UNIVERSITY CARLOS III OF MADRID

DOCTORAL THESIS

Enhancing the reliability of digital signatures as non-repudiation evidence under a holistic threat model

Author:
Jorge López Hernández-Ardieta

Supervisor:
Prof. Dr. Ana Isabel González-Tablas Ferreres

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

Leganés, February 2011

TESIS DOCTORAL

Enhancing the reliability of digital signatures as non-repudiation evidence under a holistic threat model

Autor: Jorge López Hernández-Ardieta

Directora: Prof. Dra. Ana Isabel González-Tablas Ferreres

Firma del Tribunal Calificador:

	Firma
Presidente:	
Vocal:	
Vocal:	
Vocal:	
Secretario:	

Calificación:

Leganés, de de

A Nuria

Agradecimientos

Ante todo, quiero agradecer profundamente a mi directora de tesis, Anabel, el esfuerzo que ha dedicado a lo largo de estos cuatro largos años, no exentos de imprevistos y dificultades. Anabel, no sólo has realizado una magnífica labor como directora, guiándome en momentos de incertidumbre y vislumbrando aquellas lagunas en mi investigación, sino que también me has aportado un valioso legado tras el largo viaje que supone una tesis doctoral: haber aprendido a realizar una investigación crítica, sólida y fundamentada. Gracias por tu apoyo, pero también por los momentos duros en los que tu exigencia ha hecho de esta tesis una realidad.

Quiero también agradecer a Benjamín, con todo mi cariño, su apoyo en otras lides. Te debo más de una, ya lo sabes. Pero sobre todo, te debo algo que es muy importante para mí: ser parte de este grupo de investigación, y, en definitiva, de esta Universidad. Arturo, tú también tienes gran parte de culpa, y muchas gracias por ello. Gracias también a Chema por la disposición tan estupenda que tienes siempre ante todo, y por el apoyo que me has brindado siempre que has tenido oportunidad.

No puedo expresar en palabras la deuda eterna que tendré con mis padres. Gracias a vosotros, soy lo que soy. Sin vuestra sabiduría y amor, no habría sido capaz de enfrentarme a estos retos. Y a mi hermano, que le debo tantos años de hermandad en el sentido más amplio de la palabra. Ánimo brother, que ya te queda menos para acabar la tuya.

Esta tesis es fruto de un inmenso esfuerzo que he podido realizar gracias al sacrificio de mi mujer, Nuria. Sé que sin ti no habría podido conseguirlo. Pienso recompensarte con creces durante todos los años maravillosos que nos quedan por delante.

Abstract

Traditional sensitive operations, like banking transactions, purchase processes, contract agreements etc. need to tie down the involved parties respecting the commitments made, avoiding a further repudiation of the responsibilities taken. Depending on the context, the commitment is made in one way or another, being handwritten signatures possibly the most common mechanism ever used. With the shift to digital communications, the same guarantees that exist in real world transactions are expected from electronic ones as well. Non-repudiation is thus a desired property of current electronic transactions, like those carried out in Internet banking, e-commerce or, in general, any electronic data interchange scenario.

Digital evidence is generated, collected, maintained, made available and verified by non-repudiation services in order to resolve disputes about the occurrence of a certain event, protecting the parties involved in a transaction against the other's false denial about such an event. In particular, a digital signature is considered as non-repudiation evidence which can be used subsequently, by disputing parties or by an adjudicator, to arbitrate in disputes.

The reliability of a digital signature should determine its capability to be used as valid evidence. The reliability depends on the trustworthiness of the whole life cycle of the signature, including the generation, transfer, verification and storage phases. Any vulnerability in it would undermine the reliability of the digital signature, making its applicability as non-repudiation evidence difficult to achieve. Unfortunately, technology is subject to vulnerabilities, always with the risk of an occurrence of security threats. Despite that, no rigorous mechanism addressing the reliability of digital signatures technology has been proposed so far.

The main goal of this doctoral thesis is to enhance the reliability of digital signatures in order to enforce their non-repudiation property when acting as evidence.

In the first instance, we have determined that current technology does not provide an acceptable level of trustworthiness to produce reliable non-repudiation evidence that is based on digital signatures. The security threats suffered by current technology are suffice to prevent the applicability of digital signatures as non-repudiation evidence. This finding is also aggravated by the fact that digital signatures are granted legal effectiveness under current legislation, acting as evidence in legal proceedings regarding the commitment made by a signatory in the signed document.

In our opinion, the security threats that subvert the reliability of digital signatures had to be formalized and categorized. For that purpose, a holistic taxonomy of potential attacks on digital signatures has been devised, allowing their systematic and rigorous classification.

In addition, and assuming a realistic security risk, we have built a new approach more robust and trustworthy than the predecessors to enhance the reliability of digital signatures, enforcing their non-repudiation property. This new approach is supported by two novel mechanisms presented in this thesis: the signature environment division paradigm and the extended electronic signature policies. Finally, we have designed a new fair exchange protocol that makes use of our proposal, demonstrating the applicability in a concrete scenario.

Resumen

Las operaciones sensibles tradicionales, tales como transacciones bancarias, procesos de compra-venta, firma de contratos etc. necesitan que las partes implicadas queden sujetas a los compromisos realizados, evitando así un repudio posterior de las responsabilidades adquiridas. Dependiendo del contexto, el compromiso se llevará a cabo de una manera u otra, siendo posiblemente la firma manuscrita el mecanismo más comúnmente empleado hasta la actualidad. Con el paso a las comunicaciones digitales, se espera que las mismas garantías que se encuentran en las transacciones tradicionales se proporcionen también en las electrónicas. El no repudio es, por tanto, una propiedad deseada a las actuales transacciones electrónicas, como aquellas que se llevan a cabo en la banca online, en el comercio electrónico o, en general, en cualquier intercambio de datos electrónico.

La evidencia digital se genera, recoge, mantiene, publica y verifica mediante los servicios de no repudio con el fin de resolver disputas acerca de la ocurrencia de un determinado evento, protegiendo a las partes implicadas en una transacción frente al rechazo respecto a dicho evento que pudiera realizar cualquiera de las partes. En particular, una firma digital se considera una evidencia de no repudio que puede emplearse posteriormente por las partes enfrentadas o un tercero durante el arbitrio de la disputa.

La fiabilidad de una firma digital debería determinar su capacidad para ser usada como evidencia válida. Dicha fiabilidad depende de la seguridad del ciclo de vida completo de la firma, incluyendo las fases de generación, transferencia, verificación, almacenamiento y custodia. Cualquier vulnerabilidad en dicho proceso podría socavar la fiabilidad de la firma digital, haciendo difícil su aplicación como evidencia de no repudio. Desafortunadamente, la tecnología está sujeta a vulnerabilidades, existiendo siempre una probabilidad no nula de ocurrencia de amenazas a su seguridad. A pesar de ello, hasta la fecha no se ha propuesto ningún mecanismo que aborde de manera rigurosa el estudio de la fiabilidad real de la tecnología de firma digital.

El principal objetivo de esta tesis doctoral es mejorar la fiabilidad de las firmas digitales para que éstas puedan actuar como evidencia de no repudio con garantías suficientes.

En primer lugar, hemos determinado que la tecnología actual no proporciona un nivel aceptable de confianza para producir evidencias de no repudio fiables basadas en firmas digitales. Las amenazas de seguridad que se ciernen sobre la tecnología actual son suficientes para evitar la aplicabilidad de las firmas digitales como evidencia de no repudio. Esta situación se agrava por el hecho de que las firmas digitales disponen de eficacia jurídica para actuar como medio de prueba o evidencia en procedimientos legales respecto al compromiso adquirido por el firmante en el documento firmado.

En nuestra opinión, las amenazas de seguridad que socavan la fiabilidad de las firmas digitales tenían que ser formalizadas y categorizadas. Con este objetivo, se ha diseñado una taxonomía integral de ataques potenciales a la firma digital que permite su clasificación sistemática y rigurosa.

Así mismo, y asumiendo un riesgo de seguridad realista, hemos construido una nueva propuesta más robusta y confiable que las predecesoras para mejorar la fiabilidad de las firmas digitales, reforzando así su propiedad de no repudio. Esta nueva aproximación se basa en dos mecanismos novedosos presentados en esta tesis: el paradigma de la división del entorno de firma y las políticas extendidas de firma electrónica. Finalmente, hemos diseñado un nuevo protocolo de intercambio justo donde se integra esta propuesta, demostrando su aplicabilidad en un escenario concreto.

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Introduction

This Chapter introduces the context of the thesis, the statement of the problem, the main objectives established in the thesis, the contributions achieved and the thesis organization.

1.1 Context

This doctoral thesis belongs to the area of knowledge of information security. Information security comprises the study of all aspects related to defining, achieving and maintaining confidentiality, integrity, availability, non-repudiation, accountability, authenticity and reliability of Information and Communications Technologies [111], by implementing appropriate measures or mechanisms [218].

Five basic security services are identified in the ISO OSI Reference Model Security Architecture [123]: access control, authentication, data confidentiality, data integrity, and non-repudiation. These services provide assurance against the security threats of unauthorized resource use, masquerade, unauthorized disclosure, unauthorized modification, and repudiation, respectively.

In particular, this thesis is related to non-repudiation security services. A non-repudiation service is defined by ISO/IEC as a service that generates, collects, maintains, makes available and verifies evidence concerning a claimed event or action in order to resolve disputes about the occurrence or non occurrence of the event or action [112]. The fundamental goal of a non-repudiation service is thus to protect the parties involved in a transaction against the other's false denial about such an event or action.

Non-repudiation services are of utmost importance in scenarios where parties' interests are somehow put in place and might be opposite between them, and where an unfair or malicious behavior can make a party gain a benefit from it. Unfortunately, several examples can be found, like transactions carried out in Internet banking, e-commerce, contract signing or, in general, any electronic data interchange scenario.

In order to implement non-repudiation services, non-repudiation mechanisms provide protocols for the exchange of non-repudiation tokens specific to each non-repudiation service. A non-repudiation token includes the evidence itself and, optionally, additional data. ISO/IEC defines a general model for non-repudiation mechanisms providing evidence based on cryptographic check values generated using symmetric [113] or asymmetric [114] cryptographic techniques.

Within the ISO model, a digital signature is a non-repudiation token generated using asymmetric techniques, and that is exchanged during a protocol and which can be used subsequently, by disputing parties or by an adjudicator, to arbitrate in disputes. In this thesis, we focus on the study of digital signatures when acting as non-repudiation evidence, not considering other types of evidence. This interest is supported by the fact that digital signatures are granted legal effectiveness under current legislation [76, 154, 229], acting as evidence in legal proceedings. Consequently, if a digital signature is evaluated and finally considered as valid evidence by the Tribunal, the signatory must confront the consequences derived from the signed document.

1.2 Statement of the Problem

When the life cycle of evidence is properly assured, it becomes valid in accordance with the non-repudiation policy in force. In this case, evidence is considered as proof, meaning that it serves to prove the existence of something. Valid evidence or proof avoids a later repudiation of the commitments made in the transaction by the involved parties. On the contrary, evidence which generation, transfer, maintenance or verification is not reliable cannot contribute to the establishment of proof about an event or action. It becomes useless.

A digital signature based on public key cryptography [63] fulfills the properties that non-repudiation evidence should fulfill [238]. In particular, the origin and integrity of evidence must be verifiable by a third party, and the validity of the evidence must be undeniable.

Notwithstanding, the reliability of a digital signature determines its capability to be used as valid evidence, and depends on the trustworthiness of the whole life cycle of the signature. Any vulnerability in it would make the signature lose its effectiveness.

Unfortunately, technology is subject to vulnerabilities, always with the risk of an occurrence of security threats. This situation produces undesirable consequences, that we summarize next:

- Non-repudiation evidence based on digital signatures becomes useless as there will always be a chance to prove the existence of a vulnerability in the evidence life-cycle.

- Non-repudiation evidence based on digital signatures is unfairly enforced if any of the stages within its life-cycle is compromised but it cannot be proved by the affected party.

From a legal perspective, and in case of disagreement respecting the authorship of a certain signed document, the security of the means used to produce the digital signature and evaluated at Court by an expert's report (when required) will determine whether the signature is accepted as valid evidence or not. The European and Spanish legislations have intentionally formalized three types of signatures (qualified, advanced and basic) in order to permit the application of the principle of relative importance, where the measures to protect a signature from fraud must be adequate according to the impact in case such fraud is suffered. In such cases, and taking into account the three types of signatures defined in the legislation, the next problematic scenarios can be found:

- A malicious signatory would be capable of taking advantage of the situation to generate a signature and, later on, repudiate the commitment made in the signed document by proving, on the balance of probabilities (in a civil action) or beyond reasonable doubt (in a criminal action), that a vulnerability existed in the process or the technology itself. This result would be inconsistent with the purported reliability of qualified signatures and unfair for the relying party that accepted the signature based on current legislations. There would be no entity to whom attributing the authorship of the signature, and thus the consequences derived from the signed document. The relying party would suffer from that uncertainty, which was supposed to be theoretically improbable.
- An attacker that compromised the security measures may also take advantage of the situation and generate a qualified signature on behalf of a purported signatory. If the alleged signatory was not capable of proving that a vulnerability existed in the process or technology used, he would have to deal with the commitments made in the document fraudulently signed. In this case, the alleged signatory would be clearly the one who suffered the consequences.
- In case of a basic or advanced signature, a malicious signatory would be capable of repudiating the commitment made in the signed document if the other party was not able to provide enough evidence in the opposite direction. Due to current state of technology, most likely the malicious signatory would find a vulnerability that counteracted any proof provided by the relying party. The relying party would hardly achieve having the legal effectiveness of the signature enforced by the Tribunal.

- If an attacker was able to generate a non-qualified signature on behalf of the purported signatory, and the other party was capable of providing evidence that made the Tribunal consider the signature reliable enough, the alleged signatory would have to deal with the commitments made in the document signed with the fraudulent signature. The alleged signatory would be undefended again if no evidence that proved the existence of a vulnerability or the occurrence of an attack was found.

Therefore, two unfair rulings could be made by the Court:

- The alleged signatory must deal with the commitments made in a document signed by a malicious external entity without his consent. In case of qualified signatures, it should also be mentioned that, although the alleged signatory proved that the private key was compromised, it is very likely that he will have to take on the responsibilities as he would be charged with negligence for not keeping the private key in a proper manner [60].
- The alleged signatory is capable of proving the existence of a vulnerability in the process, avoiding the signature exercising its authentication function, that is, identifying the apparent signatory as the subscriber of the document. As a result, digital signatures would be repudiable, losing their property of non-repudiation evidence, and thus, their usefulness as intended by the Law and the standards.

The reliability of a signature as evidence in a legal proceeding will highly depend on the capability to find and prove the existence of a vulnerability in the process. These scenarios are intended to raise awareness about the possible impact of applying the current legislation when the trade between the signatory and the relying party is not trustworthy due to party's misbehavior and the inherent vulnerabilities of the technology used.

The risk distribution that results from certificate revocation can be seen as a means to invalidate a fraudulent signature, freeing the owner of the signing key from any potential responsibility. However, the situation where the private key has been compromised but the certificate revocation has not been requested yet is possibly the most probable one, at least until the alleged signatory is notified about the consequences to be assumed from the fraudulently signed document.

Currently, there is little jurisprudence regarding signature repudiation, as the digital signature has not spread among end users as initially expected. Certain authors defend that the signature should be given a presumption of authenticity following the *iuris et de iure* principle, and that implies that the alleged signatory is not given the chance to refute the authenticity of the signature. In their opinion, e-commerce would be seriously damaged otherwise. However, if a signatory feels unprotected against fraud, he will not be keen on putting his interests at risk either.

In particular, we have observed the next specific problems that needed to be addressed.

P1 - Lack of formal and holistic study of the security threats on digital signatures

One must know the existence and nature of a threat, either intentional (attack) or not, before designing a countermeasure for it. Many researchers have already described and explained real or potential attacks on digital signatures, synthesizing such knowledge into formal and rigorous categorizations. However, studies undertaken for a particular field of knowledge are rarely applicable to others. In this sense, to date, no holistic and complete study of attacks or vulnerabilities specific to digital signatures has been performed. Several comprehensive and thorough studies have been done so far, but none of them can be used to study in a holistic and rigorous manner the security problem of digital signatures.

Consequently, current solutions are designed to deal with a specific subset of known attacks. Because no complete catalog of potential attacks has been devised, it is still not possible to guarantee that a holistic trustworthy solution of reliable digital signatures exists.

P2 - Current solutions are not secure enough and unrealistic assumptions are made

It has been largely demonstrated that current digital signature technology is not flawless, suffering from a wide variety of vulnerabilities that an attacker can take advantage of to subvert the security of digital signatures. In addition, the trustworthiness of current end user environments is minimum, as there is a high probability that a domestic computer is infected by malware, as reported by several studies [10, 160, 185, 198].

The level of confidence in the security measures implemented by a product can be increased by following best practices and secure software development methodologies, and performing an objective security evaluation by an accredited entity, following standard procedures [116, 120]. However, the nature of the evaluations and the unavoidable human participation in such processes only permit to obtain a level of assurance, not a certainty. Furthermore, the implied costs prevent most manufacturers from affording these evaluations.

On the other hand, the operational environment, which is of utmost importance to evaluate the overall security, cannot be easily evaluated when owned by end users.

Notwithstanding, these environments, like home PCs, laptops or mobile devices are the most common environments used nowadays by end users to perform transactions on the Internet.

Some proposals aim at enhancing the security of transactions initiated from end user environments, but few of them use digital signatures or any other type of cryptography as the transaction evidence. Taking into account ISO general model of non-repudiation and current legislation, evidence not based on cryptography lacks robustness to be considered as legally binding.

Moreover, most proposals assume the existence of a trusted platform, device or software (generally the underlying operating system). That assumption is contradictory to the problem faced: the lack of a trustworthy platform from which performing sensitive operations.

P3 - Limited electronic signature policy

ISO 13888-1 [112] establishes that, *prior to the generation of evidence, the evidence generator has to know which non-repudiation policy is acceptable to the verifier(s), the kind of evidence that is required and the set of mechanisms that are acceptable to the verifier(s).*

When using digital signatures as evidence, electronic signature policies can be used as the applicable non-repudiation policy. In this sense, a signature policy is a document that collects a set of rules to create and validate electronic signatures, under which an electronic signature can be deemed valid in a particular transaction context [71]. Thereby, transacting parties are able to determine the conditions under which an electronic signature becomes binding in a given business context.

Current definition of electronic signature policy addresses issues relating to single signatures. The signature policy defined by ETSI and IETF supports the creation and validation of a single signature. However, there are business models where more than one signature is required in order to give the transaction legal validity or to make it effective. For instance, a transaction where a contract of sale is to be signed may be considered complete if and only if the signature of both buyer and seller is present. Sometimes, the signature of an e-notary may be necessary as well. As a consequence, no signatory should be held liable until every player has made the corresponding commitment. In this case, a single signature is useless unless every signature is generated. Moreover, some non-repudiation services, like those defined in ISO model [112], may produce more than one evidence to complete a transaction, but all of them need to be linked within the same transaction scope.

The increase of paper-based processes being transposed into the digital realm makes current signature policy definition insufficient to cope with the new needs that arise.

This limitation was pointed out by ETSI in a technical report published in 2003 [72]. In particular, a transaction may need certain types of multiple signatures, that can be classified in the next three groups:

- Parallel signatures, which are applied on the same piece of information. They are mutually independent signatures where the order of the signatures is not important. For instance, a document may be electronically signed by two or more parties, like in the contract sale example given above.
- Sequential signatures, which differ from parallel signatures in that the order is significant (e.g. a data flow or transaction chain).
- Embedded signatures, where one signature is applied to another. The sequence in which the signatures are applied is important and there is a strong interrelationship. An example is a process where an electronic signature must be signed (authorized) by another (e.g. an e-notary signature applied to another).

By combining these three types of signatures, all needs related to electronic signatures can be covered. It is clear that the current signature policy definition must be extended to include the management of multiple signatures.

We consider that there is an increasing need to resolve the problems stated above. Digital signatures are considered by current legislation a key element to boost the e-commerce under secure conditions. However, end users are being the main impediment to such desired growth. They are prone to mistrust technology, specially when the commitments that derive from its use, in particular, the application of non-repudiation services, are legally binding. A reasonable and provable level of trustworthiness in non-repudiation evidence would significantly increase the confidence of users, benefiting the e-commerce in general.

The technological context within which the life cycle of a digital signature is enclosed is paramount to set the boundaries of the security problem and thus find appropriate solutions. In this thesis, we delimit the field of study to the generation and verification stages of the digital signature life-cycle, and where the participation of a user is required, as we consider that they are the most critical stages and which pose a higher risk to the signature reliability. In addition, we delimit the research to environments that comply with the functional models provided by CEN in [46] and [47], which are the reference models in the generation and verification stages, respectively.

Therefore, in this thesis we address the described problems for digital signatures generated by end users, who own and control the signing key and interact with the signing capabilities offered by the environment. We also consider digital signatures verified by end users who participate during the verification of the signed information. The physical environment where the signing or verification processes takes place can be either under the user's control and possession (e.g. personal computer, corporate laptop, mobile phone, personal digital assistant etc.) or operated by a service provider not necessarily related to or under the control of the user (e.g. any public place like a metro station, bank etc.), but in any case accessible by him.

1.3 Objectives and Contributions

The main goal of this thesis is to enhance the reliability of digital signatures, enforcing their non-repudiation property when acting as evidence.

In our opinion there was a need to address the next research topics, which have been established as the objectives of this thesis:

O1 - Perform a **formal and holistic study of the security threats on digital signatures** that permits to obtain a wide and complete view of the threats that might affect the reliability of digital signature-based evidence. This formalized knowledge would clearly contribute to design more robust approaches with appropriate countermeasures.

O2 - Propose a more **robust and realistic approach** to generate and verify digital signatures in a reliable manner and that better counteracts the threats identified in the aforementioned study. The approach must not assume the existence of a trusted platform or element on which leveraging the sensitive operation (i.e. the generation and verification of evidence). Thereby, the reliability of digital signatures under realistic scenarios will be enhanced, and thus the reliability of non-repudiation evidence, increasing the confidence of users in electronic transactions and technology in general.

O3 - **Evolve current electronic signature policy** in order to be able to manage a set of signatures generated in a single transaction. Thereby, the presence and relationships among signatures as a binding requirement to enforce non-repudiation evidence can be stipulated. The procedures to generate and validate multiple signature-based evidence according to the new policy must also be specified.

The achievement of these objectives has led to the next contributions:

C1 - A **taxonomy of attacks** covering both the signature creation and verification stages has been devised (see Chapter 6). The taxonomy has been designed to include a complete and holistic set of attack categories that permits a better understanding of the threats that may subvert the reliability of digital signature-based evidence. The taxonomy has been complemented with a method for the systematic classification of attacks on digital signatures.

C2 - An **intensive survey and classification of attacks on digital signatures** has been performed (see Chapter 10 and Appendix B). The attacks have been classified using the taxonomy and the method of classification, including both practical and theoretical attacks on the generation and verification stages of digital signatures.

C3 - The **signature environment division paradigm** has been formally presented (see Chapter 7), including a set of mechanisms for the generation and verification of digital signature-based evidence in a reliable manner. This paradigm is independent of the underlying technology or the protocol in which evidence is produced. Moreover, the proposal permits to divide the environment in as many environments as desired. Thereby, it theoretically achieves perfect security if infinite environments are used, reducing the probability of a successful attack to zero.

C4 - An **extended electronic signature policy** has been defined (see Chapter 8). Using this policy, the dependences and relationships among the signatures generated in the same transaction can be established. The policy definition has been given in ASN.1 and XML notation. Besides, the procedures to be followed by the transacting parties, both for the generation and validation of multiple signatures according to the policy, have been detailed. This extended policy supports, among other things, the practical implementation of the division paradigm, stipulating the signatures that are required to satisfy the evidence validity.

C5 - A **new fair exchange protocol** has been developed, using a design based on the signature environment division paradigm and the extended electronic signature policy. A fair exchange protocol is a protocol that ensures that no party gains an unfair advantage over the other during the protocol execution [237]. Therefore, either both parties obtain the expected information or none of them obtains any useful information from the other. Additionally, these protocols have a clear applicability in e-commerce scenarios where digital signatures are a key element [197]. This contribution is the result of two well differentiated design stages. In the first stage [95], the protocol, named OFEPSP (**O**ptimistic **F**air **E**xchange **P**rotocol based on **S**ignature **P**olicies) was conceived to oblige the parties to follow the conditions established in a signature policy (as defined by current standards [71, 203]). We considered that a first version fulfilling the basic properties of a fair exchange protocol and the novel approach of signature policies was recommended before integrating the division principle. In the second stage, OFEPSP was improved (OFEPSP+), modifying the design to comply with the division principle and our new extended signature policy definition [98]. This is protocol presented in Chapter 9.

The relationship between the problems detected, the research objectives stipulated and the contributions achieved is shown in Table 1.1.

Problem	**Objective**	**Contribution**
P1 Lack of formal and holistic study of the security threats on digital signatures	O1 Formal and holistic study of the security threats on digital signatures	C1 Taxonomy of attacks on digital signatures C2 Survey and classification of attacks on digital signatures
P2 Current solutions are not secure enough and unrealistic assumptions are made	O2 Robust and realistic approach	C3 Signature environment division paradigm C4 Extended electronic signature policy C5 New fair exchange protocol
P3 Limited electronic signature policy	O3 Evolve current electronic signature policy	C4 Extended electronic signature policy C5 New fair exchange protocol

Table 1.1: Relationship between problems, objectives and contributions

As can be seen, contribution C4 not only helps to achieve the general objective O3 but also to complete the robust and realistic approach represented by objective O2. Similarly, C5 can be considered as a contribution where contributions C3 and C4 are put into practice, fulfilling both objective O2 and objective O3.

The research results published in scientific journals and conferences during the development of the present thesis are listed in Appendix A.

1.4 Thesis Organization

This thesis consists of several chapters distributed along five different parts:

Part I. Introduction. This part introduces the whole document, and contains the present Chapter.

Chapter 1. Introduction. This is the present Chapter, and contains the thesis context, the statement of the problem, the research objectives and main contributions achieved. Also, the notation used along the thesis is given.

Acronyms, Abbreviations and Definitions. The acronyms and abbreviations used along the thesis are contained in this section. Also, the definition of some terms commonly used in the thesis are also given.

Part II. State of the art. This part analyses the state of the art that has a relevant implication on this thesis. The different themes and technologies reviewed have been organized in several chapters.

Chapter 2. Fundamentals on Digital Signatures. The fundamentals and basis for the implementation and usage of digital signatures are studied in this Chapter, including the digital signature cryptosystem, the public key infrastructure, electronic signature formats, electronic signature policies and a review of national and international legislation on electronic signature.

Chapter 3. Non-repudiation Services. This Chapter analyses the ISO/IEC non-repudiation standard, focusing on the digital signature as non-repudiation evidence. The Chapter also provides a brief review of fair non-repudiation and fair exchange protocols.

Chapter 4. Taxonomies of Attacks and Vulnerabilities in Computer Systems. In this Chapter, the efforts aimed at formalizing and classifying the computer security threats are reviewed.

Chapter 5. Security Enhancing Technologies and Methods. This Chapter comprises the most relevant technologies and proposals designed to enhance the security of computers and sensitive operations, either when digital signatures are used or not. In any case, these technologies could be the building block for producing reliable digital signature-based evidence, and as such, are analyzed from a critical viewpoint. The proposals have been classified in different sections attending to the method and strategy applied.

Part III. Proposal. This part includes the proposal elaborated to fulfill the research objectives established above.

Chapter 6. A Taxonomy of Attacks on Digital Signatures. This Chapter includes the taxonomy of attacks on digital signatures and the method of classification.

Chapter 7. Division of the Signature Environment. The signature environment division paradigm is formalized in this Chapter, covering the specific mechanisms for the generation and verification of reliable evidence under the new model.

Chapter 8. Extended Electronic Signature Policies. The policy definition that permits the management of multiple signatures within a single transaction scope is proposed in this Chapter. The Chapter also includes the generation and verification procedures that must be followed by signers and verifiers in order to comply with an extended signature policy defined for a particular transaction.

Chapter 9. An Optimistic Fair Exchange Protocol based on Signature Policies. This Chapter encloses the last proposal of the thesis: a fair exchange protocol based on the division paradigm and extended signature policies, named OFEPSP+.

Part IV. Evaluation and Conclusions. The evaluation of the thesis contributions and the conclusions are given in this part.

Chapter 10. Evaluation. This Chapter contains the evaluation of the thesis contributions:

- The evaluation of the taxonomy against the general requirements for taxonomies, and the results obtained from the intensive survey and classification of attacks on digital signatures.
- The analysis of the enhancement of the reliability of digital signature as non-repudiation evidence when using our division paradigm and extended signature policies proposal. The categories of attacks identified in the taxonomy have been used as input for the analysis. Also, the formal proofs that demonstrate the improvement of the division paradigm respecting current state of the art are also given.
- A description of the experimental implementation and evaluation of the validation algorithm of the extended electronic signature policy framework and a simulator for OFEPSP+.
- A formal analysis of the security of OFEPSP+ using automated reasoning techniques.

Chapter 11. Conclusions and Future Work. In this Chapter, the conclusions of this thesis are provided. In addition, future research work that may derive from the contributions of the thesis are outlined.

Part V. Bibliography and Appendices. This part includes the bibliography used, the scientific publications and patents derived from the research undertaken, and a number of appendices for a better comprehension of the thesis content.

Bibliography. The bibliography contains the list of references to other research papers, technical documents and standards used in the thesis.

Publications and Patents. In this Appendix, the papers published and patents filed by the author within the scope of the thesis work are listed.

Classified Attacks on Digital Signatures. This Appendix includes 112 attacks classified using the taxonomy and method of classification proposed.

Extended Signature Policy Validation Algorithm. The pseudo-code of the validation algorithm described in Chapter 8 is given in this Appendix.

Test Cases for the Extended Signature Policy Validation Algorithm. This Appendix contains the test cases used for the evaluation of the validation algorithm of the extended electronic signature policy framework. The test bench used for the test cases is also detailed.

ASN.1 and XML Schemas. This Appendix describes the ASN.1 and XML schemas for the extended electronic signature policy.

Extended Signature Policy Example for OFEPSP+. This Appendix provides an extended signature policy example written in ASN.1 that could be applied in an e-commerce transaction that uses the OFEPSP+ protocol.

High-Level Protocol Specification Language for OFEPSP+. In this Appendix, the High-Level Protocol Specification Language (HLPSL) for the OFEPSP+ protocol, and used for the formal analysis of such protocol, is detailed.

1.5 Syntax Notation

SP	Signature policy used during the generation/verification of certain digital signature-based evidence
$extSP$	Extended signature policy used during the generation/verification of certain multisignature-based evidence
$X \rightarrow Y : m$	Entity X sends message m to entity Y
$X \leftarrow Z : d$	Entity X retrieves data d from a repository located at entity Z
$S_X\,(m)$	Digital signature of entity X generated on message m
$S_X\,(m \vert SP/extSP)$	Digital signature of entity X generated on message m under SP and $extSP$ conditions
$S_X^E\,(m)$	Digital signature of entity X generated on message m within the boundaries of environment E
$S_X^E\,(m \vert SP/extSP)$	Digital signature of entity X generated on message m under SP and $extSP$ conditions and within the boundaries of environment E
$ATS_{t\_n-t\_n'}$	Time period of an absolute timing and sequence dependence (ATS). For example, $ATS_{t\_3-t\_4}$ means that the node represents a signature that must be generated between t_3 and t_4). If only ATS_{t_n} is used, then the time mark corresponds to the *NotBefore* field of the ATS *SigningTime* type. Values of t_n (*NotBefore*) and t_n' (*NotAfter*) are represented in yyyymmddhhmmss format (e.g. 19970717000000).
$\Delta(ss, mm, hh, dd)$	Maximum delta in seconds, minutes, hours and days that a relative timing and sequence dependence can have.

Acronyms, Abbreviations and Definitions

Acronym	Term
ATS	Absolute timing and sequence dependence
CS	Counter Signature
ct	commitment type
ext-SP	Extended electronic signature policy
PS	Primary Signature
RTS	Relative timing and sequence dependence
SCA	Signature creation application
SCD	Signature-creation data
SCDev	Signature creation device
SCE	Signature creation environment
sp	signature policy
SSCDev	Secure signature creation device
SSi	Set of signatures
SVA	Signature verification application
SVE	Signature verification environment
TSi	Tree of signatures

Term	Definition
Attack	Occurrence of a threat that compromises an asset or system resource by exploiting a vulnerability in the IT system. Malicious external fault that tries to provoke a service failure.
Digital signature	Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of that unit and protect against forgery e.g. by the recipient.
Electronic signature	Data in electronic form which are attached to or logically associated with other electronic data and which serve as a method of authentication.
Evidence	Information that either by itself or when used in conjunction with other information is used to establish proof about an event or action.

Non-repudiation service	Service that protects the parties involved in a transaction against the other party denying that a particular event or action took place.
Proof	Corroboration that evidence is valid in accordance with the non-repudiation policy in force.
Secure signature creation device	A signature-creation device, either hardware or software, that meets the requirements laid down in Annex III of the European Directive.
Signature-creation data	Unique data, such as codes or private cryptographic keys, which are used by the signatory to create an electronic signature.
Signature creation device	Configured software or hardware used to implement the signature-creation data.
Taxonomy	Practice and science of classification. System for naming and organizing things.
Threat	Potential violation (accidental or intended) of the security policy of a system.
Trustworthiness	Assurance that a system will perform as expected.
Vulnerability	Internal fault that enables an external fault to harm the system.

Part II

State of the Art

Chapter 2

Fundamentals on Digital Signatures

In this Chapter, the fundamentals and basis for the implementation and usage of digital signatures are explained. They cover the digital signature cryptosystem, the infrastructure that supports the usage of public key cryptography in open environments, electronic signatures as formats that define the structure and information of digital signatures, electronic signature policies, which permit to establish the requirements for a signature to be deemed valid in a particular transaction context, and a brief review of national and international legislation on electronic signature.

2.1 Digital Signatures

A digital signature is defined as "*data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of that unit and protect against forgery e.g. by the recipient*" [123].

A digital signature cryptosystem is based on public key cryptography [63], where there is a key pair that consists of a public key, publicly known, and a private key, only known by the signatory. Other configurations where the private key is stored in a central key management system for key escrow and further recovery may be implemented. However, and as stated in the thesis introduction, we only consider digital signature schemes where the private key is intended to be controlled and known exclusively by the signatory.

Using a digital signature cryptosystem, the signatory can generate a digital signature on certain data by using his private key. Afterwards, a relying party or verifier can use the public key to verify the digital signature. Therefore, a digital signature based on public key cryptography (e.g. RSA [200]) can be used to authenticate the signatory as the signature is created using means that the signatory can maintain under his sole control (the private key). In addition, a digital signature provides data integrity, ensuring that any subsequent change of the signed data is detectable.

2.2 Public Key Infrastructure

In order to use digital signatures in open environments like the Internet, it is quite advisable to have a means of identifying the signatory. Digital signatures as such are not capable of fulfilling this need. In order to bind the identity of the owner of the key pair with such cryptographic keys, another technology must be put in place.

The public key infrastructure (PKI) [57] was precisely designed to permit such binding. In PKI, a digital certificate (or public key certificate) binds a cryptographic public key with a subject identity and additional information. As a result, when the public key embedded in a certificate correctly verifies a digital signature, the relying party could assume that the subject whose identity is included in the certificate is the one that generated the signature.

PKI relies on a hierarchical architecture and a strong trust-based model. A digital certificate is issued by a Certification Authority (CA), which is the entity that asserts the binding between the subject and the public key. Depending on the procedures and policies followed by the CA to verify the subject's identity and issue the certificate (formalized in the Certification Practice Statements (CSP) [50]), relying parties will achieve a higher or lower level of trust in such binding. The hierarchical architecture is built from a root CA, which self-signs its own certificate, to end users certificates, existing as many mid-level CAs in between as necessary.

A digital certificate basically includes a serial number, which is an identifier unique within the CA scope, the subject identity, the issuer identity, the validity period, the certificate policies of applicability, usages for which the key has been authorized and the digital signature of the CA that issued the certificate.

The type of commitment that can be made by the signer depends on both the transactional context and other technical considerations. The purposes for which a certificate has been issued is one of the important technical constraints. In this sense, the key usage defines the purpose of the key contained in the certificate. PKI specifies in [57] the key usages a CA conforming to that standard must support. Next, the *keyUsage* extension to be included in a certificate is detailed:

```
KeyUsage ::= BIT STRING {
    digitalSignature        (0),
    nonRepudiation          (1), -- renamed to contentCommitment
    keyEncipherment         (2),
    dataEncipherment        (3),
    keyAgreement            (4),
    keyCertSign             (5),
    cRLSign                 (6),
    encipherOnly            (7),
    decipherOnly            (8)
}
```

Some key usages imply that the signer is bound to the signed information in a certain manner. For instance, the *digitalSignature* purpose is intended to be used in authentication services, data origin authentication services, and/or integrity services. The *nonRepudiation/contentCommitment* purpose means that the signer cannot later repudiate having performed the signature. Thus, this key usage is completely needed if the signer has to consume a non-repudiation service, or when the signature has to legally bind the signer respecting the signed data.

Cryptographic keys have a life cycle during which they are created, used, and destroyed. The same applies to digital certificates. A certificate is normally issued with a fixed validity period. Moreover, certificates can be revoked by the owner. A revocation is the procedure by which the subject terminates the validity of the certificate before the expiration date. Various circumstances may force the subject to revoke the certificate: change of name, change of association between the subject and the CA, and disclosure or suspected disclosure of the corresponding private key. Revoked certificates are usually published by the CA in a signed data structure called Certificate Revocation List (CRL). A CRL basically contains the issuer, the date of issuance (*thisDate* field), a list of revoked certificates, including, for each certificate, the serial number, the date on which the revocation occurred, (optionally) the revocation reason, and the date by which the next CRL will be issued (*nextUpdate* field).

Certificates revocation status must be accessible by relying parties in order to verify the validity of a certificate being used for a digital signature validation. CRLs can be reached by accessing the CRL distribution point, or using the Online Certificate Status Protocol (OCSP) [173].

The main limitation of CRL-based revocation methods is the elapsed time since a subject requests the certificate revocation until the CA publishes the updated CRL. This period of time, dependent on the *nextUpdate* field of the CRL, is up to the policy followed by the CA, and may vary from minutes or hours to even several days. During that period of uncertainty, the relying party cannot be sure about the status of the certificate. Solutions are limited to either waiting for the next update (grace period) or using other means that assure the freshness of the certificate status (e. g. fresh databases reached by means of OCSP protocol).

Current PKI standard defines the architectural model and the security and management services that support PKC within the PKI, including subject registration, certificate issuance, key pair recovery, key pair update and revocation requests, either following on-line protocols or off-line procedures [6]. PKI also provides the format, in ASN.1, for X.509 certificates and CRLs [57].

As mentioned before, trust in the CA and the binding between identity and public key is subject to the policies and procedures followed by the CA for its operation, included in the CSP. Critical issues include the subject registration and the key pair generation and issuance. More confidence is gained if the subject must physically present his identity to the CA or the delegated entity (e. g. the Registration Authority). Likewise, the level of trust will depend on the technical and organizational procedures followed to generate and issue the key pair to the subject, as well as the device inside which the key pair, specially the private key, is stored and protected. Hardware cryptographic devices, such as USB tokens or smart cards, provide a higher level of guarantee respecting the security of the process.

2.3 Electronic Signatures

In order to allow the exchange and verification of digital signatures, certain information must be linked to the signature, like the digital certificate that corresponds to the signing private key, the cryptographic algorithms used for the signature computation or the time at which the signature was generated, among others. This information is commonly collected into a data structure with specific format called electronic signature[1].

International standardization organizations like ETSI, IETF or World Wide Web Consortium (W3C) have delivered a number of technical standards where formats for electronic signatures are defined. These formats include basic forms of electronic signature (ES-BES) and advanced electronic signatures (AdES) that remain valid over long periods [74, 75]. AdES formats are grouped in CAdES [74, 187] or XAdES [59, 75], if defined in ASN.1 [122] or XML [232], respectively. Different AdES formats have been defined according to the additional validation information included in the electronic signature: AdES-T (with Time reference), AdES-C (with Complete validation data), AdES-X (with eXtended validation data) and AdES-A (with Archive validation data), quoted from longer to shorter validity period assurance.

[1]Electronic signature definition under current standards differs from the definition given by current legislation on electronic signatures, where an electronic signature is *data in electronic form which are attached to or logically associated with other electronic data and which serve as a method of authentication*

AdES formats that include additional validation information allow a relying party to obtain a higher assurance respecting the validity of the certificate used during the signature creation. Therefore, these formats intend to support the verification stage, but do not positively affect the reliability of the creation stage.

2.4 Electronic Signature Policies

A signature policy is a document that collects a set of rules to create and validate electronic signatures, under which an electronic signature can be deemed valid in a particular transaction context [71, 203]. Thereby, transacting parties are able to determine the conditions under which an electronic signature becomes binding in a given business context.

The policy is used by the signer in order to generate the signature according to its requirements. Afterwards, the verifier must use the policy to decide whether the signature is valid or not. For instance, a signature policy can establish that the signature must have been computed using specific cryptographic algorithms, and using a hardware cryptographic device. Besides, the same policy can also stipulate that the verifier has to timestamp the signature once previous requirements are checked.

A signature policy can be used in a wide variety of scenarios and contexts. Public and private sectors can benefit from signature policies to delimit the requirements to be fulfilled by the signatures that must be exchanged in their transactions. Furthermore, transactions in which there is a legal requirement to follow a specific form, procedure or ceremony of signing, the usage of a signature policy outlines the outer limits respecting the application and consequences of the signatures. As an example, signatures created solely for data origin authentication purposes can be distinguished from those created for content approval or authorization. Therefore, a signature policy enforces the binding feature of signatures, once the requirements that must be fulfilled by the signer and the verifier are clearly specified. If a signature has not been generated according to the policy established in the particular transaction, the commitment made by the signatory is not binding.

A signature policy is focused on the technical requirements that a signature must comply to, and, as result, the operations that both signers and verifiers must implement during the generation and validation of the signature. The association between these technical requirements and the transactional context in which the signatures are needed must be externally established. For example, a government that offers services to the citizens through electronic applications (e.g. eHealth, eTaxes, etc.) can indicate in the corresponding Web page that a certain signature policy has to be used by the citizens when generating their signatures. Furthermore, the users of the policies (signers and verifiers) must show their consent with regard to the policy terms. Contractual or other arrangements can be used to make the users recognize the binding nature of the policy.

The document that collects the rules of the policy can be written in an informal text form provided that the rules of the policy are clearly identified. A formal notation like ASN.1 [73] or XML [70] can also be used for that purpose. If only an informal text form has been used to define the policy, then some human interaction will be needed when generating and validating the signature against the policy requirements. The signer would need to read the policy and compound the signature accordingly, while the verifier would need to analyze the signature and manually evaluate the policy requirements against the actual signature information. However, by using ASN.1 or XML syntax for the policy specification it is possible to automatically generate and validate compliant signatures by using suitable programs.

Basically, a signature policy contains the following information:

- An Object Identifier that uniquely identifies the policy among those issued by the same authority.

- Information about the signature policy issuer.

- Field of application of the signature policy. It refers to legal, contractual or application contexts in which the signature policy is to be used and the specific purposes for which the electronic signature is to be applied.

- The signature policy validation section, which defines, for the signer, the data elements that shall be present in the electronic signature that is generated and, for the verifier, the data elements that shall be present for an electronic signature to be potentially valid under that signature policy. Validation information can be time stamps, Online Certificate Status Protocol (OCSP) responses, Certificate Revocation Lists (CRLs), etc.

- Constraints for generating electronic signatures: use of smartcard, use and management of attribute certificates, etc.

- Constraints for electronic signatures validity generated under this signature policy: maximum validity period for the generated signature, grace period, etc.

- Commitment type that can be made by the signer in relation to the signed data: proof of origin, proof of receipt, legal commitment, notary, witness, proof of acknowledgment, etc.

As commented in Section 2.2, the key usage field of a certificate defines the purpose of the key contained in it. However, if a binding signature or non-repudiation evidence is needed, taking into account key usage only seems to be insufficient. As allowed by RFC 5280, a certificate can be issued for more that just one purpose, though the standard restricts the combinations of bits that may be set in an instantiation of the key usage extension. As a result, this extension could include both *nonRepudiation* and *keyAgreement* bits set to '1'. The signer could allege that he intended to use his certificate for establishing the SSL/TLS connection with the seller's Web site, rather than issuing a purchase order.

Signature policies serve as the means to establish higher level of security requirements for the signatures of a transaction, filling the gap left by the key usage. The signer can select the desired commitment type, and include it as a signed attribute of the signature [74, 75]. Moreover, the signer must indicate in the electronic signature which policy it complies to. For that purpose, a reference to the signature policy has to be included as a signed attribute as well. The reference consists of the unique policy identifier (OID), the hash value of the policy and the hash algorithm identifier used to compute the hash value over the signature policy description. If a dispute arises, the policy can be used to provide supportive evidence over the procedure associated with a specific signature in use. Implicit references to the policy can be made as well but the structure or semantics of the document that a user is signing must be well defined.

It is worth noting that the signature policy reference is a signed attribute inside an electronic signature format [59, 74, 75], and therefore it is not possible to substitute the policy used during the signature generation without invalidating such signature. This prevents the possible situation where an attacker wants to limit the requirements under which the signature is considered to be valid by replacing the policy by a less demanding one.

As mentioned in [71, 203], the field of application and semantics fields define the specific use and meaning of the commitment within the overall field of application defined for the policy. As can be noticed, commitment types that can be defined in a signature policy differ from those defined by RFC 5280 in the key usage extension. The latter are straightforward and explicit, while, in the former, a more flexible approach has been followed. Notwithstanding, a conceptual relationship between both sets exist. A signature intended to bind the signer with the signed content should use a certificate with the *nonRepudiation* bit set to '1'. In the same way, the commitment type chosen from those available in the signature policy should cover that intention (e.g. content approval). Though not every combination of key usages and commitment types are coherent, a matrix of correspondence or consistent combinations could be traced. However, as commitment types of a signature policy are context specific, this matrix should be implemented in a case-by-case basis.

An important contribution of signature policies is that they enforce enclosing the consequences that can be derived from a signature. Used in conjunction with appropriate key usages, the signer can obtain a certain level of confidence respecting the type of signature that is being generated.

However, the increase of paper-based processes being transposed into the digital realm makes current signature policy definition insufficient to cope with the new needs that arise. Very often, documents require more than one signature to give it legal validity or to make a transaction effective. This limitation was pointed out by ETSI in a technical report published in 2003 [72]. ETSI report studies business needs that may need multiple signatures, and provides a foundation for further work in relation to the technical implementation of a signature policy governing multiple signatures, and a general guidance on a methodology for the validation of multiple signatures. ETSI report assumes that each signature will be validated under a signature policy for single signatures such as [73] or [70]. The challenge raised by ETSI is the specification and validation of the relationship of each required signature against the others. At the time of writing this thesis, no technical solution covering this need had been proposed.

2.5 Electronic Signatures from a Legal Viewpoint

The electronic signature has become a key element in the information society. Several national and international legislations recognize the legal effectiveness of electronic signatures and their admissibility as evidence in legal proceedings. In addition, current legislation specifically grants electronic signatures an important role for promoting e-commerce under secure conditions [68, 76, 154, 184, 229]

An electronic signature is defined in art. 2 of the European Directive on electronic signatures as "*data in electronic form which are attached to or logically associated with other electronic data and which serve as a method of authentication*" [76]. Authentication should be understood as a means of identifying the signatory but also indicating the signatory's approval of the signed data, as interpreted in the incorporation of the Directive into the national laws by most Member States [60]. The UNCITRAL definition of electronic signature [229] also supports this statement: "*Electronic signature means data in electronic form in, affixed to or logically associated with, a data message, which may be used to identify the signatory in relation to the data message and to indicate the signatory's approval of the information contained in the data message*".

Consequently, the electronic signature is functionally equivalent to the handwritten signature. The signatory is legally bound respecting the commitments made in the signed document once his knowledge and approval of the content of the document are consciously represented by his electronic signature. The electronic signature acts as instrument of evidence regarding the authenticity of the electronic document in the same way as the handwritten signature does regarding the paper-based document.

The legislation on electronic signature analyzed herein is technology-neutral, as the technical or procedural requirements for generating and verifying electronic signatures are not specified [40]. They establish generic requirements that must be fulfilled by the implementing technology, either present or future [76, 154, 184, 229]. An electronic signature which conforms to these requirements (functional equivalence) will have legal effect, no matter its nature or technical background. This model grants to market forces the power to decide what constitutes an electronic signature.

In the Spanish Law [154] and the European Directive on electronic signatures [76], three types of electronic signatures are defined, each one providing a different degree of reliability. A definition for the (basic) electronic signature has already been given above. An advanced electronic signature requires that it is uniquely linked to the signatory, it is capable of identifying the signatory, it is created using means that the signatory can maintain under his sole control, and it is linked to the data to which it relates in such a manner that any subsequent change of the data is detectable. Finally, a qualified signature is mainly an advanced one generated using a certificate that complies with Annex I of the European Directive on electronic signatures, and a secure signature creation device.

In this sense, and according to the European Directive on electronic signatures [76], a signature creation device (SCDev) is *configured software or hardware used to implement the signature-creation data* (SCD), being the SCD *unique data, such as codes or private cryptographic keys, which are used by the signatory to create an electronic signature*. A SCDev, either hardware or software, that meets the requirements laid down in Annex III of the European Directive is called a secure signature creation device (SSCDev). Annex III dictates that the SSCDev *must, by appropriate technical and procedural means, ensure at least that (a) the signature-creation-data used for signature generation can practically occur only once, and that their secrecy is reasonably assured; (b) the signature-creation-data used for signature generation cannot, with reasonable assurance, be derived and the signature is protected against forgery using currently available technology;(c) the signature-creation-data used for signature generation can be reliably protected by the legitimate signatory against the use of others.* Therefore, SSCDev places more confidence regarding the protection of the SCD.

On the other hand, the Canadian Act on Personal Information Protection and Electronic Documents [184] establishes the same requirements for a "secure electronic signature" as those given for an advanced signature under the European Directive, while the UNCITRAL Model Law on Electronic Signatures [229] establishes similar requirements for an electronic signature to be reliable.

Based on the current state-of-technology, only cryptographic digital signatures satisfy the aforementioned requirements for advanced or qualified signatures. In this sense, some legislations explicitly state that a digital signature supported by a Public Key Infrastructure (PKI) is one of the potential underlying technologies. For instance, the European Directive refers to digital certificates and signature creation devices, and the UNCITRAL Model Law even establishes PKI and digital signatures as an example of implementing technologies for generating compliant signatures.

A qualified electronic signature is legally equivalent to a hand-written signature, complying with the formal requirements established for the latter. The European Directive indicates that these electronic signatures "*satisfy the legal requirements of a signature in relation to data in electronic form in the same manner as a handwritten signature satisfies those requirements in relation to paper-based data*". Notwithstanding, electronic signatures not considered as qualified are legally recognized as well according to the European Directive and the Spanish Law. Article 5.2 of the Directive says that "*Member states shall ensure that an electronic signature is not denied legal effectiveness and admissibility as evidence in legal proceedings solely on the grounds that it is in electronic form, or not based upon a qualified certificate, or not based upon a qualified certificate issued by an accredited certification-service-provider, or not created by a secure signature-creation device*".

The main difference between qualified signatures and the rest is the potential evidential value. In legal proceedings, a qualified electronic signature, under the provision of the European Directive and within the European boundaries, is given a favorable validity judgment a priori (*ex ante*) as it is considered the most reliable form of signature. On the contrary, the basic and advanced electronic signatures need a posteriori (*ex post*) judgment by the Court, as their reliability is assured to a lesser extent [60].

That means that, if it is proved that the signature is a qualified one, the alleged signatory must provide evidence that questions, beyond reasonable doubt, its security in case its authorship, and thus of the signed document, is repudiated. The onus of proof is legally reversed, moving the burden of proof to the alleged signatory instead of to the verifier [167]. In case of a basic or advanced signature, the alleged signatory is also capable of repudiating the authorship but it is the other party the one who must provide evidence that support the reliability of the signature.

Some authors make the assumption that the qualified signature is given a *iuris tantum* presumption of authenticity [60], that is, the alleged signatory is given the chance to provide evidence that contradicts the established presumption of authenticity of the signature. The situation would be different if the presumption of authenticity followed the *iuris et de iure* principle, supported by other authors, and that implies that the alleged signatory is not given the chance to refute the authenticity of the qualified signature. Here, the alleged signatory is automatically assumed to be the actual signatory, having to deal with the consequences always. For example, the UNCITRAL Model Law on Electronic Commerce [229] indicates that a receiver is entitled to regard a data message as being that of the originator, and to act on that assumption, if, in order to ascertain whether the data message was that of the originator, the receiver properly applied a procedure previously agreed to by the originator for that purpose (e.g. use of certain type of electronic signature). Also, the Digital Signature Guidelines of the American Bar Association [107] refers to the equitable principles as a means to not give the signatory the chance to repudiate the signature. In the Spanish context, the unclear presumption of authenticity given in article 3.8 of 59/2003 [154] for qualified signatures has been clarified in the Spanish Law 56/2007 [154], granting the *iuris tantum* approach.

If the technology used to produce the electronic signature is based on digital signatures, and a digital certificate issued within a PKI is owned by the signatory, then the certificate revocation could limit the liability derived from such signature. Four possibilities exist regarding the phases inherent in the revocation process [175][1]:

- The private key has been compromised but the certificate revocation has not been requested yet. It may be either because the owner does not suspect of the key compromise or, if he did, he has not sent the revocation request to the Certification Authority (CA) yet. In this case, since the owner of the certificate is obliged to securely keep the signing key, he would have to take on the responsibilities. Art. 22 and art. 23 of 59/2003 [154] exonerate the CA from any liability if the owner did not guarantee the confidentiality and access to the signing key diligently, or did not notify the key compromise when suspected. As commented by Cruz in [60], the principle of risk distribution has its own particularities in the electronic signature field, and under the Spanish legislation. It is mentioned that, due to the strong security measures granted to a qualified signature, the judge could assume that the signing key must have been compromised only due to a

[1]It should be mentioned that this analysis was performed by Nadal et al. on the Spanish "Real Decreto 14/1999 de firma electrónica", dated from 1999, and that preceded the current 59/2003 Law. Therefore, some conclusions given therein may not be accurate respecting the content of 59/2003.

negligent behavior of the owner. It is suggested that the alleged signatory would have to either be bound to the document content (attribution of data message) or deal with the damages caused to the relying party. However, this conclusion will have to be inevitably achieved by the Tribunal on a case by case basis.

- The owner has sent the revocation request to the CA, but it has not been processed yet or, though processed, the new status has not been made available yet. The inevitable uncertainty period between the revocation request and the moment at which such revocation is made effective shifts the risk assumption to the CA, according to art. 22.3 of 59/2003.
- The revocation status has been made effective and published, but the relying party is not capable of accessing it. If the reason is related to the scheduling for the actual publication of the certificate status, then the liability will be assumed by the CA. Otherwise (e.g. the relying party had no connection to the publication service at certain moment but due to an internal problem), the liability will be assumed by the relying party.
- The revocation status is updated and made available to any relying party. In this case, the certificate revocation list is updated and accessible. Therefore, art. 23.4 of 59/2003 shifts the liability to the relying party.

Chapter 3

Non-repudiation Services

This Chapter briefly reviews the ISO/IEC model for non-repudiation services, specifically those based on digital signatures. Also, two particular types of protocol, named fair exchange and fair non-repudiation protocols, and that use non-repudiation services, are explained.

3.1 General Model for Non-repudiation Services

According to ISO/IEC 13888-1 [112], a non-repudiation service protects the parties involved in a transaction against the other party denying that a particular event or action took place. Non-repudiation services permit to design protocols and applications where strong commitments are made between the participant entities. Electronic commerce protocols or e-Government services are among those scenarios that must protect the entities against fraud and misbehavior. In non-repudiation services, digital evidence permit to enforce the responsibility that each entity takes on in the transaction, avoiding a further successful repudiation of the commitments made.

Non-repudiation services are built based on non-repudiation mechanisms that provide protocols for the exchange of non-repudiation tokens specific to the service. Because there is a wide variety of commitments that can be made by the transacting parties, so are the non-repudiation services that have been standardized. According to [112], the next non-repudiation services may be provisioned:

Non-repudiation of creation, intends to protect against an entity's false denial of having created the content of a message.

Non-repudiation of sending, intends to protect against the sender's false denial of having sent a message.

Non-repudiation of origin, intends to protect against the originator's false denial of having created the content of a message and of having sent a message, covering both non-repudiation of creation and non-repudiation of sending.

Non-repudiation of receipt, intends to protect against a recipient's false denial of having received a message.

Non-repudiation of knowledge, intends to protect against a recipient's false denial of having taken notice of the content of a received message.

Non-repudiation of delivery, intends to protect against a recipient's false denial of having received a message and recognized the content of a message, covering both non-repudiation of receipt and non-repudiation of knowledge.

Non-repudiation of submission, intends to provide evidence that a delivery authority has accepted a message for transmission.

Non-repudiation of transport, intends to provide evidence for the message originator that a delivery authority has delivered a message to the intended recipient.

In [112], ISO/IEC defines a general model for non-repudiation mechanisms providing evidence based on cryptographic check values generated using symmetric [113] or asymmetric [114] cryptography techniques. Under this model, evidence is generated, collected, maintained, made available and verified by non-repudiation services in order to resolve disputes about the occurrence or non occurrence of a certain event or action. Evidence is information that either by itself or when used in conjunction with other information is used to establish proof about an event or action. Proof is the corroboration that evidence is valid in accordance with the non-repudiation policy in force. Though evidence does not necessarily prove the truth or existence of something, it contributes to the establishment of such proof. On the other hand, the non-repudiation policy is whatever set of criteria for the provision of the non-repudiation service, that is, the set of rules to be applied for the generation and verification of evidence and for adjudication. In particular, ISO/IEC 13888-1 establishes that, *prior to the generation of evidence, the evidence generator has to know which non-repudiation policy is acceptable to the verifier(s), the kind of evidence that is required and the set of mechanisms that are acceptable to the verifier(s).* Thereby, and when using digital signatures as evidence, electronic signature policies, as defined in Section 2.4, can be used as the applicable non-repudiation policy.

Zhou [238] formerly established the specific properties that non-repudiation evidence should fulfill. In particular, the origin and integrity of the evidence must be verifiable by a third party, and the validity of the evidence must be undeniable. As commented by Zhou in [238], non-repudiation is related to authentication but has stronger proof requirements.

A non-repudiation token, which is the information actually exchanged in a particular non-repudiation service, includes the evidence itself and, optionally, additional data. ISO/IEC general model defines three types of non-repudiation tokens:

Generic non-repudiation token (GNRT), which can be used in many non-repudiation services.

Time-stamping token (TST), generated by a Time-stamping Authority (TSA), and which establishes evidence regarding the time at which the token was generated. This token is needed in case the clock provided by the entity generating evidence cannot be trusted.

Notarization token (NT), generated by a notary authority, and which provides evidence about the properties of the entities involved and of the data stored or communicated.

During the provision of a non-repudiation service, at least one non-repudiation token derived from the GNRT must be generated. Additionally, TST and/or NT can be generated and used to verify such token.

A trusted third party (TTP), which is an entity trusted by the entities involved in the service provision, may be necessary during the protocol execution. For instance, TST and NT are tokens that need a TTP for their generation. The degree of participation of the TTP during the evidence exchange varies, existing TTPs that intervene in every message transmission (inline), only in certain transmissions (online), or just when a protocol interruption or entity misbehavior occurs (offline) [115]. Depending on the type of evidence being produced and the TTP involved in the non-repudiation service, the evidence generation, transfer, storage, retrieval and verification phases differ.

3.2 Non-repudiation Using Digital Signatures

A digital signature is defined by ISO/IEC 13888-1 [112] as *data appended to, or a cryptographic transformation of, a data unit that allows the recipient of the data unit to prove the source and integrity of the data unit and protect against forgery e.g. by the recipient.* In [114], a digital signature is further defined as a non-repudiation token generated using asymmetric techniques, and that is exchanged during a protocol and which can be used subsequently, by disputing parties or by an adjudicator, to arbitrate in disputes.

Digital signatures supported by a public key infrastructure can actually behave as non-repudiation evidence as they guarantee [1]:

- A binding between the public key used to verify the signature and the identity of the signer, by means of a digital certificate issued by a trusted authority. Thereby, the origin of the evidence (authenticity) is verifiable by the relying party.
- Due to the cryptographic properties of digital signatures, any modification of the signed information or the signature itself is detected, assuring the integrity of the evidence and the signed information.
- The originator of the signature (the signer) cannot later repudiate the authorship, since he is the only one that (theoretically) knows the private key.

The digital signature must be verified in conjunction with the signed information, the validity of the signature must be satisfied, and the digital certificate must be valid at the time the signature was computed. When these requirements are met, the evidence is considered as valid. Consequently, a digital signature acting as non-repudiation evidence and that is correctly verified according to the particular non-repudiation policy is suffice to resolve a possible dispute, avoiding the alleged signatory to successfully repudiate the commitment made in the transaction.

A digital signature acting as non-repudiation evidence can be generated:

- By the evidence subject, that is, the entity responsible for the action, or associated with the event, with regard to which evidence is generated.
- By a TTP on behalf of the evidence subject or an evidence requester.
- By a TTP on its own. In this case, the TTP could be the evidence subject (e.g. when acting as a delivery authority) or an entity acting as a notary or monitoring authority that provides evidence about the entities involved and data stored or communicated between the entities, or evidence about what was monitored, respectively.

In the same way, a digital signature acting as non-repudiation evidence can be verified by any of the next entities but using the public key certificates and certificate revocation lists which were all valid at the time the evidence was generated:

- By the evidence user, that is, the entity that uses non-repudiation evidence.
- By a TTP acting as evidence verification authority, and which is invoked by the evidence user.

[1] Chapter 2 provides detailed information about digital signatures and related technology.

In case of identity-based signatures, the public system parameters needed to verify the signature can also be obtained from the trusted authority along with the signer's identity.

It should be mentioned that advanced electronic signature formats (AdES) that include the signature policy reference (see Chapter 2) and a time reference (AdES-EPES-T) are suffice to fulfill the requirements imposed by ISO model on non-repudiation for generic non-repudiation tokens (GNRT) [112] and non-repudiation tokens specific to the service being provided (non-repudiation of origin tokens (NROT), non-repudiation of delivery tokens (NRDT), non-repudiation of submission (NRS) tokens and non-repudiation of transport (NRT) tokens) [114]. Therefore, an electronic signature with such format can be considered as a non-repudiation token that includes non-repudiation evidence. The purpose of such evidence will depend on the commitment type to which the signer adheres (see Section 2.4), and the non-repudiation service to be provided.

3.3 Fair Non-repudiation and Fair Exchange Protocols

The growth of the e-commerce has allowed companies and individuals to sell and purchase almost any kind of product and service through the Internet in a fast, comfortable and effective manner. The main operational modes are B2C (Business to Customer) and B2B (Business to Business). In the former case the buyer is an individual while in the latter is another company. Although the context is different, in both cases the aim is the same: to purchase a product or service (resource).

Such a purchase implies an electronic transaction. Next, an example of the stages needed for a B2C transaction is described:

1. First of all, the buyer selects the resource to buy.
2. Later, the buyer has to send his credit card information to the seller.
3. During next stage a payment gateway is used for communicating with buyer and seller's banks, and for carrying out the charge process [69, 211].
4. Finally, the seller notifies the buyer about the transaction result.

Once the transaction is finished, and depending on the purchase conditions and the resource nature, the buyer obtains either the resource itself or an acknowledgment of receipt. As an example for the first case, purchasing stream content may allow the buyer to start receiving the resource as soon as the transaction is successfully completed. In the second one, the acknowledgment of receipt is the only element the buyer has for making a future complaint. This acknowledgment of receipt acts as a proof of the performed electronic transaction. In e-commerce context, this acknowledgment of receipt may be an electronic invoice.

However, possessing certain information does not always imply a contractual or legal commitment. It is possible for the buyer to auto-generate acknowledgments of receipt or even for the seller to charge as many purchases as desired, once buyer's credit card information is known. Thereby a buyer could reject having participated in an electronic transaction or the seller could not send the resource to the buyer if he suspected that a fraudulent operation was carried out.

Evidence is used for resolving this previous problematic situation. Evidence is generated during the transaction, and obliges both buyer and seller to make a certain commitment. Additionally, evidence built using electronic signatures are granted legal effectiveness, acting as an instrument of evidence in legal proceedings, as analyzed in Section 2.5. For example, in [11], evidence is used in several e-commerce protocols proposals. In one of these protocols, the buyer generates evidence on the order and bank account information, while the seller does it on the acknowledgment of receipt. Due to the nature of the evidence, and according to current standards on non-repudiation, the commitment cannot be successfully repudiated. Therefore, the buyer makes a commitment to paying the agreed price for the resource while the seller makes it to the resource delivering. And no one could, in a future dispute, successfully reject having made those commitments.

Nevertheless, the mere evidence generation does not completely resolve the problem. Due to the division of an electronic transaction into several stages, as seen in the B2C example above, the seller could obtain the evidence from the buyer without sending the corresponding one. For being a complete fair exchange process, evidence must tie down both buyer and seller on an equal footing. Thereby, none of them could gain an advantage over the other during the protocol execution.

Several solutions have been proposed in the literature to address fair exchange in e-commerce [197]. These protocols are known as fair exchange protocols. Asokan, in his thesis [11], undertakes a profound research on fairness and non-repudiation in e-commerce, reviewing the properties of these protocols and focusing on the design of a generic payment service. In [197], several fair exchange protocols are analyzed, even from the early gradual exchange protocols. On the other hand, Bao et al. [23] propose a new cryptographic primitive called CEMBS (Certificate of Encrypted Message Being a Signature), and from which different fair exchange protocols are built. Conditional digital signatures are proposed by Lee and Kim [149] as the key element of the fair exchange protocol design.

Fair exchange protocols also try to assure the timeliness property [11]. Strong timeliness is defined by [188] as the property by which, "*at any moment in an ongoing* *protocol run, an honest party P can be sure that the protocol will be automatically completed at a certain point in time. If any action is required from P, it should be clearly stated, as well as the circumstances in which it should be taken. At completion, the state of the exchange is either final or any further changes to the state will not degrade the level of fairness achieved by P*".

The design of fair exchange protocols slightly differs from those known as fair non-repudiation protocols [51, 142, 234, 237], where the exchanged information is, in fact, the non-repudiation evidence. While in a fair exchange protocol both buyer and seller know the items to be exchanged before executing the protocol (e.g. e-commerce scenario), in a fair non-repudiation protocol the recipient of a message does not expect a particular message, knowing it only at the end of the protocol [142].

However, both fair exchange and non-repudiation protocols share a common aspect: the existence of an entity trusted by both players and that participates during the protocol for assuring the fairness and timeliness. This entity is called Trusted Third Party (TTP), and, as demonstrated by Pagnia and Gärtner in [181], it is impossible to achieve fairness without a TTP. Depending on the degree of involvement of the TTP during the protocol execution, it is considered inline, online or offline. Offline TTPs improve the performance of the protocol in normal executions. Protocols that incorporate an offline TTP are called optimistic [12, 141, 179], and assume that both players will not misbehave, requiring the participation of the TTP only when this assumption is broken by any of the players.

Chapter 4

Taxonomies of Attacks and Vulnerabilities in Computer Systems

This Chapter reviews some relevant taxonomies and formal classifications of attacks and vulnerabilities in computer systems proposed along the last decades. Those studies focused on digital signatures are of special interest, although others are also covered if useful information can be extracted from them. On the contrary, many taxonomies specific to certain areas of knowledge, like intrusion detection systems, network security or Web servers have not been considered, since they do not provide useful information for the purpose of this thesis. A brief introduction to the taxonomy concept is given first. The remainder of the Chapter focuses on proposals coming from the academia and expert bodies.

4.1 Introduction

Taxonomy, from the Greek axis ('order', 'arrangement') and nomos ('law' or 'science'), is the practice and science of classification. A taxonomy or taxonomic scheme is *a system for naming and organizing things, especially plants and animals, into groups which share similar qualities* (Cambridge Dictionary).

A taxonomy is useful for the classification of the knowledge (specimen) of a particular field. Thereby, by applying a systematic and rigorous analysis, that knowledge can be classified into a set of well defined categories. From a general viewpoint, the benefits of a taxonomy are multiple. A taxonomy permits splitting a complex phenomenon into more understandable pieces of information. As a result, a taxonomy makes further studies possible, providing a common agreed base, and identifying the parts of the phenomenon that are less known. And, using the classification of the taxonomy, one is more capable of explaining observed phenomena.

Several taxonomies targeting the security area in computer science have been proposed during the last years [106]. The fields of interest vary, covering Intrusion Detection Systems [18], vulnerabilities and computer programs flaws [35, 36, 143, 186], software attacks at application level [146], computer security intrusions [155], network security [90], attacks on secure devices [196], and many others.

Two perspectives can be chosen when approaching security threats with a taxonomy. A perspective focused on the vulnerability exploited in the attack or another one based on the attacker's method. Since software vulnerabilities comprise the majority of security problems a computer system may have, analyzing the causes of software vulnerabilities can provide a tool to software developers to build more robust applications and systems. On the other hand, understanding the methods used by the attackers to exploit vulnerabilities permits to implement adequate security countermeasures, so a taxonomy of attacks may be useful as well.

4.2 The Dimension Approach: Lindqvist and Johsson's Taxonomy and Others

A dimension is a property or attribute that permits a classification of an attack or a vulnerability to take a more holistic view of such an event. Each vulnerability or attack can be split into several properties. Each property is then used to classify such an event from a different perspective (e.g. source of the attack, method of the attack, result of the attack, etc.), all of them complementary as a whole. Each author has chosen different dimensions according to the goal of the taxonomy and the approach followed for the classification.

Lindqvist and Johsson introduced the concept of dimension in their early paper [155] to classify computer security intrusions. In particular, Lindqvist and Johsson make a classification from the system's owner viewpoint, focusing on the external observations of attacks and breaches which the system owner can make. Consequently, they define two dimensions, called intrusion techniques and intrusion results, each of which is refined in different categories and subcategories.

The dimension concept is similar to the characteristics approach used by Landwehr et al. in [143] to classify computer program security flaws according to the genesis, time of introduction and location of the flaw.

In [34], Bishop performs a taxonomy of Unix system and network vulnerabilities, introducing the concept of axis, which conceptually follows the same approach as a dimension. In his taxonomy, the vulnerabilities are classified attending to six axes:

nature of the flaw, time of introduction of the vulnerability, exploitation domain, effect domain, minimum number of components necessary to exploit the vulnerability and source of identification of the vulnerability. In [36], Bishop carries out a critical analysis of the studies and taxonomies of vulnerabilities proposed till the mid '90s. Bishop's work is interesting since he raises awareness respecting the difficulty of designing a well-formed taxonomy, specially respecting the mutually exclusive and non-ambiguity requirements that a taxonomy should fulfill.

Howard and Longstaff designed a process-driven taxonomy where multiple factors (attackers, tool, vulnerability, action, target, unauthorized result, objectives) are used for classifying security incidents [103]. A fixed number of categories is given for each factor. An incident is thus classified according to the category selected in each factor.

More recently, Lough [158] used dimensions in his taxonomy named VERDICT (Validation Exposure Randomness Deallocation Improper Conditions Taxonomy), proposing four characteristics to classify an attack: improper validation, improper exposure, improper randomness and improper deallocation.

4.3 CEN's Classification of Threats on Signature Creation Applications

CEN CWA 14170 stipulates in [46] a detailed set of security requirements and recommendations for signature creation applications (SCA) that generate advanced electronic signatures by means of a hardware signature-creation device (SCDev).

The requirements are grouped attending to the functional component of the SCA that shall implement them. Each requirement is derived from the analysis of the corresponding security threat, for which a title and description are given.

CEN document provides an exhaustive work that covers an extensive list of the attacks on the signature generation stage. However, the analysis is limited to advanced signatures generated by means of a hardware SCDev, not including potential attacks on signatures generated with software SCDevs, and which have legal effectiveness as well.

Unfortunately, the general guidelines provided by CEN in the analog document for electronic signature verification applications [47] does not follow the same approach. CEN CWA 14171 does not include any reference to potential attacks on the verification stage, leading to an incomplete categorization of threats on digital signatures.

4.4 Hill's Taxonomy of Attacks on XML Signatures

A taxonomy of attacks on XML signatures is given in [102]. The goal of the taxonomy is to enumerate and categorize specific attacks on signatures that follow the XML signature and encryption standard. The author identifies seven categories of attacks, attending to the applied method: C14N Denial of Service (with one subcategory), transform injection (further split into three subcategories), hash collision attack against *signedInfo* with "C14N with comments", external reference attacks, reference complexity, element wrapping attacks and untrusted keys. The study also considers four attack surfaces, including canonicalization, reference resolution, key resolution and signature evasion.

Each attack category is described in terms of the involved attack surface, the impact of the attack, a textual description of the attack, an explanation of the exploit scenario and possible countermeasures.

Though the taxonomy is useful to prevent these specific attacks, it fails to provide a holistic view of attacks on digital signatures. It should be noted that none of the attack categories is focused on subverting the reliability of the signature generation. Also, the specificity of the attacks makes that other types of signatures (e.g. raw digital signatures, ASN.1 signatures, PDF signatures) fall out of the scope of the study. Moreover, it is not explained how an attack should be classified according to the taxonomy.

Additionally, most attack categories are aimed at achieving a denial of service attack, and thus not undermining the signature reliability but the signature verification capability. Others are conceived as a former attack from which executing secondary attacks, like arbitrary code execution, and consequently do not pursue subverting the signature verification either.

4.5 Kain's Taxonomy

Kain proposed in his Master Thesis [131] an outline of taxonomy for dynamic content attacks on electronic signatures. The taxonomy contains three main high-level categories: hidden parameters, fraudulent content and nature of change. Each analyzed attack is classified according to a subcategory from each high-level category.

Four different subcategories of attacks are outlined according to the hidden parameters the attacker could use to construct malleable documents that are to be digitally signed: time, viewer data, viewer action and remote control.

Kain suggests two different types of attacks depending on the moment when the fraudulent content is established by the attacker. In a pre-signature attack, the alternate content must be fixed at the time the signature is applied; in a post-signature attack, the alternate content may be chosen at some point after the signature has been applied.

Finally, nature of change category includes three subcategories: static content, dynamic content, and, finally, dynamic content and signature. A static content attack is an attack where viewing alternate content does not change the working object. On the contrary, a dynamic content attack requires the working object to be modified as part of displaying alternate content. It is only effective when the signature is verified against the original object. The last subcategory, dynamic content and signature, implies that the signature is changed at the same time the content of the object does, possibly by shipping pre-established object-signature pairs.

As mentioned by the author, the taxonomy is not complete. The author exclusively focused on attacks that can vary the semantics of the signed document. Therefore, many other types of attacks on electronic signatures are not considered in his study.

4.6 Rae and Wildman's Taxonomy

In [196], Rae and Wildman propose a matrix-based taxonomy for attacks on secure hardware devices. The matrix is dimensioned by the access required by the attacker and the action taken by the attacker. The access required is refined in four categories graded from the gravest scenario to the softest one: the attacker can manipulate the device at will, including subjecting the device to sophisticated scanning, or even modifying the device; the attacker can handle the device, manipulating the environment (e.g. inputs to the device) but without breaking tamper seals; the attacker can approximate the device, monitoring external characteristics, but cannot touch it; and, finally, the attacker can just communicate with the device through available network interfaces. On the other hand, the action dimension is split into four categories: recover a key, defeat authentication, avoid authentication and deny service.

Once a cell of the matrix is selected, the attack is further analyzed based on the consequence of the attack and method used for the attack. As a result, four properties are used to classify attacks under their taxonomy.

This taxonomy is interesting and broad regarding the security threats that may affect a signature creation device like a smart card. One of the attacks covered in this taxonomy, called side-channel attack, has been extensively studied in the literature [224]. However, due to the heterogeneity and dependencies such attacks have on the internal algorithms, the device attacked and the technique applied, no complete taxonomy has been presented so far. Other researchers have studied attacks and vulnerabilities in smart cards that may contain cryptographic material, such as private keys for signing purposes [84]. Notwithstanding, such a specialization, though necessary, implies that many other attacks on digital signatures are not covered in this type of taxonomies.

4.7 Hansman and Hunt's Taxonomy

Blended attacks are attacks that target several vulnerabilities simultaneously. As a result, a classification of a blended attack is difficult if a strategy different than a dimension-based one is followed in the taxonomy [91].

In [91], Hansman and Hunt provide a specific dimension-based taxonomy aimed at dealing with this sort of attacks, and which was intended to be the first taxonomy that gave a holistic approach to classify attacks, taking into account all parts of the attack. The authors also analyze different designs of a taxonomy, concluding that a dimension-based taxonomy is preferable to other designs based on trees or flat-lists.

Hansman and Hunt's taxonomy uses four dimensions. The first one covers the attack vector and behavior. The second dimension focuses on the attack targets. Multiple entries can be selected from this dimension, as an attack may have multiple targets. The third dimension deals with the specific vulnerability that allows the attack to be carried out. Like in the second dimension, more than one vulnerability can be selected for the same attack. The authors mention that entries in this dimension are usually Common Vulnerabilities and Exposures (CVE) entries [55], though others may be created. The last dimension classifies attacks having payloads or effects beyond themselves.

By selecting a category belonging to the first dimension in the fourth one, the authors intended to permit the classification of attacks that launched other attacks. However, the iterative process is not possible with their method of classification, and thus only a "second round" with less information than the first attack can be described. As a result, blended attacks are, in practice, hardly classifiable under their taxonomy.

Furthermore, in our opinion, providing an exhaustive list of vulnerabilities is useless in a taxonomy, since a vulnerability is always specific to a certain version of a piece of software, and taxonomies should provide solutions of general applicability. Projects like the CVE [55] or the National Vulnerability DataBase [177] give detailed descriptions of known vulnerabilities. The information is publicly available in the form of a catalog of vulnerabilities. However, no comprehensive schema or classification taxonomy is usually given. On the contrary, search tools consist of keywords search or alphabetic ordering.

4.8 The Common Attack Pattern Enumeration and Classification

The Common Attack Pattern Enumeration and Classification (CAPEC) [52] is an initiative developed by MITRE Corporation and sponsored by the Department of Homeland Security of the United States of America. CAPEC collects a large set of attack patterns. An attack pattern is defined as an abstraction mechanism for describing how a type of observed attack is executed. It is the description of a common method for exploiting software from the attacker's perspective. The attack patterns contained in CAPEC databases are generated from in-depth analysis of specific real-world exploit examples, and are publicly available.

CAPEC contains five different views, where the most interesting one is the Methods of Attacks View. In this view, the attack patterns are classified in accordance to the method followed by the attacker. In particular, the information concerned to an attack pattern includes a summary of the pattern, an attack execution flow, the prerequisites that must exist for the attack to be executable, the methods/vectors of the attack, the attacker skills or knowledge required, solutions and mitigations, related Common Weakness Enumeration (CWE) [56] and Common Vulnerabilities and Exposures (CVE) [55], and confidentiality, integrity and availability impact, among others.

Chapter 5

Security Enhancing Technologies and Methods

Since attacks on computers began to appear, researchers have been proposing solutions to counteract them or at least minimize their impact. In this Chapter, a critical review of the state of the art of relevant security enhancing proposals is given. Some proposals are explicitly focused on the signature generation problem while others were designed for other purposes, though they may be of applicability to enhance the reliability of digital signatures as well. The proposals have been classified attending to the method and strategy applied. Others with no similarity with any category have been collected in Section 5.10.

5.1 Security Assurance by Objective Evaluation

An objective evaluation comprises the procedures to assess that an IT product and/or the underlying operational system and environment reduce all security risks identified by risk assessment as unacceptable to a level that can be tolerated as residual risks. As a result, the strength of the implemented security mechanisms must suffice to counter the identified threats taking into account the expected attack potential.

The Common Criteria (CC) [53, 116] provides a common framework for the evaluation of the security functionality of IT products. The CC provides a common set of security functional requirements [117] that can be used by the developer to specify the security functionality implemented by the product. Additionally, the CC also provides a set of security assurance requirements [118] to be fulfilled by the developer during the design, implementation and test of the product, and by the evaluator during the security evaluation process, established in the Common Methodology for Information Technology Security Evaluation [54, 119]. This evaluation process provides confidence, in terms of Evaluation Assurance Level (EAL), in that the security functionality of the product and the assurance measures applied to it meet the established requirements.

Higher levels of assurance place more detailed requirements on the content and presentation style of evidence to be provided by the manufacturer. In the same way, higher assurance usually requires increasing rigor of analysis of the evidence by both the developer and the evaluator. Notwithstanding, the achieved level of assurance is a grade of confidence respecting the product security, not a certain level of security. Obviously, the higher the level of assurance is, the more confidence is gained in its actual security. However, the cost and complexity of the process considerably increases with the level of assurance to achieve, so few manufacturers conduct this evaluation or merely achieve low levels.

Within the CC terminology, a Protection Profile (PP) is an implementation independent statement of security needs for a particular type of product. A PP serves as the document where end users, organizations, governments, etc. define the security requirements to be fulfilled by the type of products the PP is targeting. Later on, a manufacturer can put a product under CC evaluation, and in which the requirements of a certain PP of interest will be considered. Particularly, four PPs have been defined in Spain for electronic signature related products, two of them establishing the security assurance requirements for an EAL1 [190, 192], while the other two for an EAL3 [191, 193]. They are oriented to signature creation and validation applications where the Secure Signature Creation Device (SSCDev) being used corresponds to the Spanish electronic identity card (eDNI). Notwithstanding, their design could be reused for any type of SSCDev conforming to the Protection Profile described in [45].

PPs [190] and [191] focus on applications with exclusive control over the interfaces that interact with the signer (e.g. PDA, smart phone, etc.) while [192] and [193] consider that the application is running in a general-purpose platform that provides those interfaces (e.g. a Personal Computer). In the latter, the platform provides the hardware and supplementary software layers (e.g. operating system, driver to communicate with the SSCDev, external applications running with user or root privileges, etc.). The document to be signed can be provided by external entities or by the signer herself by using the interfaces available in the platform.

However, these PPs assume that the platform is trustworthy, and must be configured and managed accordingly:

"*The vulnerabilities that are exploitable through the environment of the application [...] must be eliminated by means of an appropriate platform configuration and a correct usage. How the platform must be configured in order to not permit an attacker compromise the assets herein described is a hard task, out of the scope of this PP*".

In a nutshell, the PP authors do not include the security requirements that the platform on which the signature application has to be executed must fulfill. In CC terminology, it belongs to the environment, on which can be made as many assumptions as desired. Based on the EAL considered by those PPs and the assumptions made, a signature application would only be certified according to the requirements therein contained considering a secure underlying platform and a low attack potential. It is clearly a not realistic scenario.

These PPs also include some security functional requirements for the secure representation of the document to be signed, what includes detecting any active, hidden or malicious dynamic content inserted in the document. These requirements are intended to fulfill the What You See Is What You Sign (WYSIWYS) property [207]. WYSIWYS is a security measure that provides the signer with a last step verification by means of a graphical or textual representation of what is going to be signed. Once the signer confirms it, the displayed information is the one supposed to be sent to the signature creation device (e.g. smart card) and therefore the information on which the digital signature is computed. However, several attacks have been demonstrated effective to defeat WYSIWYS state-of-the-art countermeasures [8, 130, 131]. This property, normally enforced by showing a confirmation message to the signer, can be easily defeated unless the environment assures a secure underlying platform and the application implements highly sophisticated security measures.

Finally, and in order to avoid malicious usages of the PIN (Personal Identification Number) linked to the SSCDev, a security functional requirement contained in those PPs obliges the signature application to unbind it after each signing process. However, PKCS11 standard [189], one of the widest APIs used to access cryptographic devices (e.g. SSCDev), does not permit the unbinding procedure required by the PPs.

CEN has defined the security requirements that signature creation and signature verification applications should fulfill [46, 47], like in the aforementioned PPs, and which could be taken as reference in an objective security evaluation. However, those requirements can only be fulfilled if the underlying platform is fully trustworthy.

The National Institute of Standards and Technology (NIST) has also published a set of requirements for obtaining assurances for digital signature applications [24]. It provides recommendations for domain parameter validity assurance, public key validity assurance, private key proof-of-possession, and assurance of the private key's owner identity. Yet it does not take potential attacks on the cryptographic implementation or the application into account. It is only concerned with the correctness of the protocols involved and the configuration of the cryptographic implementation.

The security of a signing process depends not only on the security of the product itself, but also on its configuration and the security of the operational environment.

The operational environment includes the operational system and related technical elements, and the non-IT aspects, such as the organizational policies and procedures for managing and operating the system. As a result, a complete security evaluation should comprise all these aspects. In this sense, the CC product evaluation is limited to the verification of the security capabilities implemented by the product, so the specific operational context is not considered. However, and as stated in [157], if the underlying operating system, which is commonly shifted to the environment under CC evaluations (except obviously, when the product being evaluated is the operating system itself), is not secure, no application running on top of it can be made secure either.

In this sense, the recently published ISO 19791 [120] focuses on the evaluation of the operational environment, and is based on a three-layer model:

- Risk assessment, to determine the security risks applicable to a system (e.g. ISO 27005 [121]).
- Risk reduction, to counter or mitigate security risks by the selection, application and assessment of security controls. It should be mention that a risk can be mitigated, but never eliminated.
- Accreditation, to confirm that the residual risks remaining within the system after the controls are applied are appropriate for the system to be used in live operation.

ISO 19791 concentrates on stage two, whereas the accreditation stage is normally carried out by National Authorities following well-known procedures [86, 87].

Evaluations of the operational environment are common in back-office systems, managed either by public or private institutions, and corporate networks. Yet this type of evaluation is hardly applicable in environments managed by end users.

Evaluation processes are of utmost importance to gain an objective level of assurance of the security of product, backed by accredited and competent entities. However, it should be clear that certainty is never achieved, and, in any case, the highest level of assurance does not imply perfect security. This reality applies to every solution proposed in further Sections, and each time a security assumption is made either on the signature application, the operational environment or the communication channels.

5.2 Secure Software Development Methodologies

The security of IT products must be conceived from the very beginning. In this sense, secure software development life cycle (SSDLC) methodologies integrate specific tasks in each stage of the software life cycle in order to cover the security needs during the whole development. There is a wide number of efforts and approaches for the implementation of SSDLC [62]. SSDLC methodologies typically include the next activities. In the first instance, the regulations and rules that may impose certain security requirements in relation to the context are analyzed (e.g. PCI DSS, electronic signature legislation, etc.). An analysis of security requirements and attack use cases is also performed, and by which the abuse scenarios can be modeled. During the design phase, a threat modeling is developed, along with a system security architecture by which the security mechanisms to implement are put in place and their relationships established. A key issue during the application of SSDLC is the integration of best practices during the software coding. Using automatic static code analysis tools also helps to discover vulnerabilities in the source code. And, finally, specific security tests are usually carried out.

SSDLC methodologies significantly contribute to the achievement of trustworthy technology, and research in this direction are a must. A representative example comes from the manufacturer of the most widely deployed operating system. Microsoft is reporting 60% fewer infections in Vista operating system, released in 2008, than Windows XP, released in 2002 [169].

However, the increasing complexity of systems and the huge effort that imply deploying and integrating these methodologies into software manufactures processes make these initiatives less effective than initially expected. The counterexample comes from Microsoft again. We continuously see how new security patches have to be applied to the Windows operating system. The reduction in the number of vulnerabilities due to the application of SSDLC methodologies has not reached a sufficiently effective level yet, and much work has still to be done in this field.

5.3 Revocation Mechanisms and Limitations on Key Usages

Standard procedures that allow a legitimate owner of a digital certificate to react against the private key compromise are mainly limited to two procedures. On the one hand, certificate revocation mechanisms [6], if such compromise is known before the attacker uses the private key, and, on the other hand, legal procedures at court, if the attacker has already taken advantage of the potential benefits derived from the signed document.

Certificate revocation can also be improved when used in conjunction with grace and cautionary periods. ETSI signature policy report [71] contains the next description of the cautionary period:

"*[...] a signature is only valid after a minimum time frame has elapsed after the signature time before the signature can be relied on as legally valid. This minimum* *time frame is counted from an upper limit of the signature time [...] and is called the cautionary period. A signature before this time has elapsed can be considered conditionally valid. It may be deemed valid only once this time has elapsed and when the status information at the end of the cautionary period indicates that none of the certificates from the certification path is revoked. As a result this places time requirements on the instant when the revocation status information has to be fetched*".

Thereby, a signature is only valid after a minimum time frame has elapsed after the signature time before the signature can be relied on as legally valid. Consequently, a signature before this cautionary period has elapsed can only be considered conditionally valid. It may be deemed valid only once this time has elapsed and when the status information at the end of the cautionary period indicates that none of the certificates from the certification path is revoked.

Whether the relying party accepts a revocation request made at a time before and/or after the signing time depends upon the particular policy to be enforced. However, this solution lacks of applicability since the user must, at least, suspect that the key has been compromised. Sometimes it does not happen until the clauses of the signed document have been put in practice, being too late for a certificate revocation.

A standard mechanism for preventing or mitigating the fraudulent usage of a certificate is to define certificate extensions, supported by the specific certificate policy [50]. These extensions can impose additional constraints for the usage of a specific certificate, both in context and in transactional attributes (e.g. maximum amount of transfered money, maximum number of transactions per day, etc.). Despite these fields are restrictively informed, and due to current computers' capacity, an attacker can still cause significant damage by reusing the key as many times as allowed for the maximum amount of money permitted.

In order to cope with potential forged signatures, some authors propose the capability to revoke the signature made [199], by which the purported signer repudiates the authorship of a signature and thus the consequences that may arise from the signed document. Thereby, digital signatures are not regarded as non-repudiation evidence anymore but *plausible evidence.* Michael Roe analyzed in his thesis [201] the inherited problems of non-repudiation services as those that provide irrefutable evidence concerning the occurrence or non-occurrence of an event or action. In [201], the concept of *plausible deniability* is proposed, which permits an entity to deny its participation in a disputed event or action, as there be no irrefutable evidence that can provide certainty of the events happened or actions taken. However, usefulness of digital evidence can be severely limited if its recipient bears the risk of a later denial, which he cannot influence [165].

Zhang et al. suggested a model where both signer and relying party collaborate to revoke the signature [236]. It was not a unidirectional revocation, but an agreed one.

In the same direction, the concept of conditional signature was presented by Lee and Kim in 2002 [149]. In this case, the signatures exchanged during an electronic transaction between the origin and the receiver are valid provided that certain conditions are fulfilled. These conditions are directly included in a new digital signature schema proposed.

Berta et al. designed a framework based on conditional signatures for mitigating the consequences of using an untrusted terminal when creating digital signatures [27]. This framework requires the use of a trusted terminal - they cite a home PC - from which confirming the validity or invalidity of the signature. However, this requirement goes against the principle of signature revocation, that is, the impossibility to have the means for creating reliable signatures. More importantly, this proposal does not provide any detail respecting the procedure the user must follow in order to confirm or revoke a signature at the trusted terminal. Considering that the user has to request the smart card to show the operations performed (translated to a set of commands sent to the card interface), and taking advantage of the fact that the smart card cannot be linked to a specific terminal, an attack could be mounted on their protocol. In particular, the attacker could initiate the revocation procedure on behalf of the user (steps 5 to 8 in Section 4 of [30]) at the untrusted terminal. The attacker could revoke authentic signatures providing that the conditional timeout has not elapsed and the smart card is still accessible. This is possible if the revocation request is performed by the attacker immediately after the signature computation. In [29], they comment the possibility of configuring the protocol with a default deny approach, by which a signature would remain invalid after the timeout unless it is explicitly confirmed by the user before that time. Following the same strategy, a more dangerous attack could be mounted to automatically confirm fraudulent signatures at the untrusted terminal.

Another detailed proposal for signature revocation is given in [152], where an XML-based design is described. The proposed framework allows the signatory to claim a revocation of a previously made declaration of will contained in a signed document. The revocation method consists of producing a signed revocation token, uniquely linked to both the signatory and the electronic signature. Several causes may derive in a signature revocation. While some of them are not related to attack reasons (e.g. revocation of a declaration or annulment of a legal act/contract induced by erroneous dispatching of the declaration or duress, mutual annulment of a contract, etc.), and permit the signer to compute the revocation token by means that remain under his "absolute" control, others derive from a compromise of the signing process (e.g. private key compromise, cryptanalysis attacks or semantic attacks). In this case, the existence of a trustworthy computer for the generation of the revocation tokens is deemed necessary, and suggests us the question why not using it for performing the signature itself. Secondly, in order to revoke a signature the signatory must realize that an attack was performed. As already commented, maybe it is too late for preventing the attacker to derive a benefit from it.

In general, the signature revocation concept intrinsically implies the existence of a trustworthy platform from which revoking or confirming previously generated signatures.

5.4 Smart Card-based Solutions

Smart cards are considered trusted tamper-proof devices that securely store and use cryptographic material. Besides the fact that there are direct attacks on these devices [41, 61, 77, 80, 84, 147, 208, 223], smart cards do not have a direct interface to the user, and thus need to leverage in the untrusted environment most of the signing processes.

In order to reduce the number of operations to be executed in the untrusted environment, Kilian-Kehr and Posegga provide a set of protocols where the most part of the signing process is shifted from the untrusted components into the trustworthy element of the platform: the smart card [135]. In one of the protocols therein proposed, the card must store the signature application and be able to process and hash the whole document (feasible depending on the document's size). Besides, the card must receive the document to be signed from the originator through a dedicated channel, which usually has a constrained bandwidth. The main contribution of the paper is a protocol where the signed information is derived from an interaction with a service provider (e.g. the case of an e-commerce transaction). In order to protect the signing process, the protocol needs a trusted computing platform that executes certain script code for the document-to-be-signed generation. In this case, the whole security of the process depends on this assumption. Moreover, input from the user is not protected, what would allow an attacker to modify the actual sensitive information of the transaction.

In [28], Berta proposes an interesting solution to allow the user to generate reliable messages by using multiple smart cards at an untrusted terminal. The message to be signed is split into several pieces of information (signals) with no semantic meaning (as the attacker can modify the content) except for the smart card that signs each signal. The message meaning is thus extracted from the sequence of signed signals. The protocol seems secure against potential attacks. However, the proposal lacks of real applicability. If the signal is bit-based, signing a long message demands an incredible amount of time from the user, besides requiring him to manually translate the message into binary code (using automatic means would imply that the attacker has a chance to modify the meaning of the message). On the other hand, creating a set of encoding rules (e.g. each card represents a number/set of letters, a word, ...) makes the protocol less flexible. The more abstract the encoding rules are, the less number of semantically feasible messages can be sent.

5.5 Usage of Mobile Devices

Mobile devices have been widely proposed in the literature as the means to provide a trustworthy environment for digital signature computation [204]. In particular, Personal Digital Assistants (PDA) and latest generation mobile devices have reasonable computational capacities and advanced user interfaces. As such, they are a candidate for enforcing the reliability of the signing process.

In [132], the authors propose a PDA to act as a personal device for controlling a smart card attached to it using an asymmetric key pair for creating digital signatures, and circumventing the problems involved with untrusted document viewers. The same philosophy is used in [161], where a PDA is used again for computing digital signatures.

Another solution, named Trusted Pocket Signer, funded by the German Federal Ministry of Economics and Labor, is presented in [92]. This solution consists of a PDA which combines secure wireless communication, trustworthy visualization of documents to be signed, smart cards and user authentication based on biometrics as realization of the willful act in order to provide a trustworthy signature creation environment.

Nevertheless, every mobile-based solution makes some wrong or dangerous assumptions. In most cases, the underlying platform of the device is considered trustworthy, or at least, more resilient to current attacks. Quite the opposite, in 2004, a research on the security of mobile operating systems concluded that they were not suited to produce legally binding signatures, as none of them supported secure data management and there were still a large number of open security gaps in those operating systems [172].

The increasing use of this type of device for daily life communications has made attackers target them as a profitable victim. To date, there are several potential attacks that can subvert the security of mobile devices such as those used for digital signatures [108]. Recently published surveys [170, 217] have analyzed the main threats for mobile devices, and it can be seen that attacks on mobile networks and devices are growing in number and sophistication every year.

As a result, a solution exclusively based on mobile devices does not provide an added value, suffering from the same types of threats as traditional platforms.

5.6 Forcing a Confirmation Step

In the mid 90's, an innovative payment system for e-commerce that used a confirmation channel was proposed by First Virtual Inc. [110]. In order to make a transaction initiated with a merchant effective, the buyer had to confirm it by replying an email sent by a secure server of the payment system. In the authors' opinion, attacks that could compromise the user's computer (without being traced) for monitoring incoming emails to confirm fraudulent transactions (or reject authentic ones) were considered too sophisticated and with a low probability of occurrence. Nowadays, it has been proved that multiple types of attack can achieve that.

Another related payment system followed the same philosophy, but, instead of requiring a confirmation by email reply, it included a second authentication round when needed (e.g. once the user was correctly authenticated after providing the correct password, a challenge-response mechanism could be executed by the so called payment switch) [83].

Solutions that create a confirmation channel (email, HTTP, etc.) rooted at the same platform where the transaction is initiated lack of effectiveness against current threats. If the platform and operational environment is under the attacker's control, the attacker can subvert the confirmation channel at its will.

Some home banking solutions incorporate a confirmation step where the user must enter a code previously sent to his mobile phone in the Web site where the transaction has been initiated (e.g. to transfer funds). Nevertheless, no strong evidence is generated (and of course no evidence fulfilling the non-repudiation property). The confirmation action does not bind the user in any sense, once the code can have been intercepted by an attacker, or even an untrusted Web site could have committed the transaction on behalf of the user. In cases where the confirmation step is performed in an environment different than the one that originated the transaction, a split trust paradigm is being put in practice.

5.7 Split Trust Paradigm

The split-trust paradigm or model was introduced in 1999 by Balfanz and Felten [22], and claims that hand-held computers (mobile devices such as PDAs or smart phones) are more suitable for carrying out sensitive operations like digital signatures or credentials management. In contrast to solutions that uniquely employ a mobile device, reviewed in Section 5.5, under this new paradigm multiple devices work in tandem. Applications are split into two parts: one part that runs on a hand-held, trusted but resource-limited device; and a second part that runs on a more powerful and capable platform, normally untrusted (e.g. a PC, laptop or public terminal). Both parts work in conjunction to offer the user certain functionality (e.g. secure Web browsing, operation with public terminals, generation of digital signatures, etc.).

Several authors propose the application of split trust model to protect the user's privacy and security when operating with public or other type of untrusted terminals [81, 166, 180, 202, 215, 216].

Ross et al. [202] provide a detailed architecture that allows the user to complete a transaction initiated at an untrusted terminal by using a trusted device for the security-enhanced service interactions. However, the user is not allowed to enter data at the trusted terminal, being restricted to the visualization of sensitive service content only. A similar philosophy has been recently followed by Maurer and De Luca in [166], but in this case, the user is allowed to enter sensitive data from the mobile device.

The same perspective has been applied in [215] to a secure Web Browsing context, using a single commercially available cell phone as the trusted personal device. The user must confirm the transaction initiated at an untrusted terminal by using the trusted device, being able to enter PINs, password or whatever sensitive data are needed. The data flow from the trusted terminal to the Web server through the untrusted terminal (via bluetooth or USB connection, for instance), but encrypted with a key previously agreed between both endpoints. Each time a transaction is initiated, the user will notice it by receiving the message on his mobile device.

Sharp et al. describe in [216] a system that enables users to access their applications and data securely using a combination of public terminals and a more trusted, personal device. Thereby, the sensitive and private information is censored at the public terminal, being accessible only at the trusted, mobile device.

In [81], the user leverages a personal mobile device to establish trust on a public terminal prior to revealing personal information to that terminal. The user employs his mobile device to determine the identity and integrity of the software installed in the public terminal, which must contain a Trusted Platform Module (TPM) [226] and the Integrity Measurement Architecture (IMA) [206] to support software attestation. Though this solution is focused on public terminals, a signatory could apply it to verify the state of his home PC before computing a signature. Notwithstanding, the authors recognize that several attacks are possible, including malware that compromises the state of the terminal during runtime (after the attestation has been performed), and a Reboot-between-Attestations attack, by which the terminal could reboot and run malicious software after attesting to its software integrity but before the user reveals personal data.

In [168], the authors propose a system that uses a trusted mobile device as a proxy between a keyboard and a TPM-equipped host platform (e.g. home PC) to establish a trusted channel for sensitive user input to applications. The aim of the system is to avoid malware running at user level on the host platform from capturing the user's input (e.g. a PIN or password). Therefore, the considered attack potential is drastically reduced, not covering malware with root privileges or that could compromise the operating system kernel. The authors also assume that the user's host platform is capable of attesting to its current software state, using a TPM architecture. Otherwise, a compromise of the active kernel would permit an attacker to capture the sensitive input. In addition, an attack on the mobile device is not considered either. Otherwise, the whole process is vulnerable. Furthermore, the software components installed on the host platform whose integrity is verified by the TPM architecture are restricted to the boot stack, the kernel, its modules and well-ordered system services. Finally, the system is not resilient to hardware keyloggers.

The solution given in [168] does not avoid a malware from compromising the host platform providing that it does not violate the integrity of the verifiable software components. Other runtime attacks can be performed by the malware for compromising the sensitive data managed by the trusted application during its operation. Thus, vulnerabilities of the trusted applications can still be exploited by the malware. In general, a signing process over a document stored in the untrustworthy host platform can also be compromised. It should be noted that this solution is focused on protecting the user's input to trusted applications and providing a trusted visual output to the user. As a result, most of the threats that may arise in the context of this thesis are not counteracted by design.

A specific solution for enhancing the reliability of digital signatures is presented in [129]. Jøsang and AlFayyadh propose a protocol for ensuring the WYSIWYS property by requiring two different platforms for completing the signing process. The user selects the document to be signed in the Document Processing Platform (DPP). DPP must transmit this document to a Portable Signature Platform (PSP) with cryptographic capabilities. Afterwards, an image of the screen where the document is being displayed must be captured by the user with a Digital Camera integrated in the PSP. This image is processed by an OCR (optical character recognition) software, composing a digital copy of the document selected by the user. If the document transmitted to the PSP by the DPP and the document generated by the OCR in the PSP itself are equal, then the user can perform the signature of the document by using the PSP. Finally, the signature is sent back to the DPP and verified against the original document. A patent application with the same approach was filed by Müller-Quade et al. [174]. In this case, a prototype was implemented in Java Micro Edition for mobile phones.

Though the approach found in [129, 174] mitigates the signature environment untrustworthiness problem, it has usability constraints. But more importantly, a malware that compromised the PSP could generate binding signatures on behalf of the user on whatever information. These signatures could act as non-repudiation evidence with malicious purpose though the protocol was not completed.

The aforementioned solution proposed by Berta et al. [27] can be classified as a split trust solution as well, though the trusted computer is not a mobile device but a PC platform. The signatory has to make use of a trusted computer to confirm or revoke the signatures generated at an untrusted terminal.

5.8 Trusted Computing Technologies

Trusted Computing Technologies intend to provide computers that consistently behave in expected ways, being those behaviors enforced by hardware and software and using cryptography techniques.

Balacheff et al. presented a method to increase the trust in open computing platforms, such that a signatory can be confident when generating digital signatures [21]. In particular, their method requires the platform to be equipped with a TPM and a Trusted Display Controller (TDC), and the user must own a smart card. The architecture described enforces a secure display by shifting the control of the video graphics processing to the TDC. There is also a secure communication channel between the TDC and the smart card, assuring that what the user is visualizing on the screen corresponds to the data sent for the signature computation. In a nutshell, the solution therein proposed expects to assure the WYSIWYS property of a signing process. They claim that their solution avoids the need to trust in the underlying platform, as the whole signing process is carried out by reliable components. However, their proposal requires a specific hardware architecture, including a TPM and a protected video circuitry. In addition, the authors assume that the user has access to a trusted platform for loading certain information (more specifically, the seal image) into the smart card, and that is needed for the subsequent trustworthy signature computation.

In [220], Spalka et al. explain how to protect the creation of digital signatures against Trojan Horse programs by combining the Intelligent Adjunct Model defined by Balacheff in 2000 [19] with a TPM. In the Balacheff's model, an adjunct (the smart card in their proposal) is given control over off-card resources, becoming an active element that can initiate transactions. Spalka et al. elaborate three scenarios in which the signing software location differs: a scenario where the signing software is installed in the PC (classical view); a second one where the smart card sends the software to a Java

Virtual Machine for the signing operation; and a third scenario where the smart card directly communicates with the user, and thus keeps the signing software on its own.

The architecture given by the authors in [220] incorporates a SWORM (Software Write Once Read Many) medium which would allow the signer to securely write the document to be signed into it for a later signature computation. Afterwards, the SWORM medium would transmit the document to the signature application, according to its location (the scenarios previously mentioned). They assume that a Trojan Horse trying to forge a signature will only react when the document is marked as signable. Therefore, they consider that the malware will not attempt to alter the data during its transmission from the application that generates the document (e.g. a standard editor) and the SWORM medium. To us, this assumption is questionable, since a malware can be specifically designed to undermine a concrete security mechanism like this, and thus could perform a man-in-the-middle attack between the application and the SWORM medium.

Initiatives and technologies coming from the Trusted Computing Platform Alliance (TCPA) [20] pave the way for achieving more trustworthy platforms. However, TPM-based approaches are still vulnerable to certain attacks, like the TOCTOU attack (Time-of-check Time-of-use) [38], where a malware can subvert the integrity of the platform once it has been verified during the boot process. Thereby, there is still a window of vulnerability during which an attacker can compromise an application or even the underlying operating system before the signature is generated. More attacks can be carried out on a TPM, like a reset attack (needs physical access to the platform), a BIOS attack, a Bootloader attack or the cold-boot attack, as explained in [171].

Although we consider that TCPA initiatives are paramount, they do not completely eliminate the probability of a successful attack while at the same time extra complexity and cost are added to end user computers.

5.9 Server-Aided Signatures

Server-aided or server-assisted signatures is a trend in which the signature computation is delegated to an external server managed by a Trusted Third Party (TTP), and which acts on behalf of the user [33, 64, 156, 194]. Thereby, users that own resource-limited devices still have the possibility of generating digital signatures in mobile-based transactions.

While some protocols permit the user to send the document or message to be signed to the server, others like [135] (a variant of the Kilian-Kehr et al. protocol explained in Section 5.4), obliges the document to be stored as a Web resource in the TTP. As a result, the signatory's capability to sign local documents is eliminated.

The main disadvantage of server-aided signatures is that, as the signature computation is delegated to an external entity, the signatory loses the control over the signing means, and thus, the generated signatures could not be attributed to the signatory (see Chapter 2 for further details). Furthermore, the server becomes the main target of the attacks. If the security of the TTP is compromised, then the cryptographic material of every user can be accessible by the attacker.

5.10 Other Proposals

There are some proposals classified under the terms human-computer cryptography [163] or visual cryptography [176], and that try to enforce secure authentication or encryption mechanisms making use of end user capabilities only. As a result, these proposals do not rely on external cryptographic devices such as smart cards. However, they suffer from severe drawbacks, as remarked by Berta and Vajda in [31]. For instance, if the user is not able to encrypt and decrypt messages in one step, then the remote partner is not able to help the user to establish a secret channel either. Or if the user is unable to calculate an authenticator that cannot be broken by the terminal, then the remote partner cannot help the user in constructing an authenticated channel.

Maurer studied the intrinsic limitations of digital signatures and the related liability issues [165]. In his opinion, there must be a trade-off between the usefulness of digital signatures acting as digital evidence and the digital liability exposure derived from that evidence. As a signature can be generated without the user's consent or knowledge (e.g. malware, ambiguous interfaces, etc.), he proposes to support the liability by producing a digital declaration consciously generated by the signatory. As a consequence, the liability of a signature would rely on the user's awareness at the time of generating the signature. Maurer argues that, though the user still has the possibility to deny the digital declaration, it implies a consequence as serious as denying the authorship of a hand-written signature, being forced to testify. Therefore, the digital evidence acts as additional evidence that discourages the purported signatory to falsely deny having generated the corresponding digital signature. However, and contrary to the author's opinion, such digital declarations can also be forged by an attacker. Consequently, digital declarations do not provide additional security to the signing process.

A method specifically focused on solving the problem of signing digital documents with dynamic content is given in [8]. Therefore, this proposal merely counteracts a very specific subset of attacks. The method relies on Microsoft Component Object Model (COM) architecture. The program that composed the document to be signed is invoked by the signature application in order to parse the document and eliminate all dynamic content. In addition, this program must be shared by both the signer and the verifier, but the COM approach can resolve the issue by making the program available through a registered COM object in Windows environment. The security analysis performed by the authors rises several attacks on their schema. On the other hand, the authors state that their countermeasure cannot be effective if the attacker gains access to the signer's or verifier's computers, and is able to violate the integrity of certain data (e.g. the extension/application association tables).

In [144], a method to ensure the integrity and authenticity of client-server communication from end to end is presented. This solutions could be applied to remote transactions where a user had to make a binding commitment, like home banking transactions. The authors assume that the user is not logged in the system as an administrator and no malware is running with administrator privileges. Furthermore, the integrity of the operating system on which the trusted path implementation runs must be assured. Because the authors do assume the existence of malware, it is not ridiculous to think that it could compromise the operating system too. As they remark in their paper, if the underlying platform has vulnerabilities, one should refrain from using any security-relevant software on it.

Buccafurri and Lax implemented a solution based on Java Cards aimed at strengthening the reliability of the signing process [42]. The signer uses a Java Applet loaded from the Java Card to check whether the document sent to the card by the signature software has been modified by a malware. The Java Applet reads the file and sends it to the Java Card for a second hash computation acting as a check round. However, and as admitted by the authors, their solution is effective only against malware running at user level, more specifically, malware that has compromised the signature software. A malware performing a man-in-the-middle attack between the operating system I/O interfaces and the Java Applet could easily cheat the security mechanism. On the other hand, the solution only targets a single type of attack, being vulnerable to many others.

Part III

Proposal

Chapter 6

A Taxonomy of Attacks on Digital Signatures

In this Chapter we propose a taxonomy that contains a holistic categorization of attacks on digital signatures, covering attacks on both the generation and verification phases. The Chapter also includes a method for the systematic classification of attacks according to the taxonomy.

The Chapter is organized as follows. The terminology used along the remainder of the Chapter is given in the next Section 6.1. The models for the signature creation and signature verification environments are discussed in Section 6.2. The threat model from which the taxonomy will be devised is detailed in Section 6.3. The taxonomy of attacks on digital signatures is described in Section 6.4. Section 6.5 provides the method to systematically classify an attack on digital signatures according to the taxonomy. Finally, we conclude the Chapter in Section 6.6.

6.1 Terms and Definitions

Before affording the definition of a taxonomy, the terms and concepts applicable to the particular field of knowledge must be fixed. The terminology contained in this Section is based on widely accepted terminology on information security and dependability. In particular, the definitions are based on [17], where Avizienis et al. presented a profound work that defines and classifies basic concepts of dependable and secure computing. Other definitions covering digital signatures and specific related technology were given during the analysis of the state of the art, in Part II.

A **system**, from a broad viewpoint, is an entity that interacts with other entities (i.e. other systems, including hardware, software, humans) and the physical world with its natural phenomena. In this thesis, our system corresponds to, on the one hand, the signature creation system (see Section 6.2.1) and, on the other hand, the signature verification system (see Section 6.2.2). The specific definitions that apply to those systems are given in the corresponding Sections.

The **function** of a system is what the system is intended to do and is described by the functional specification in terms of functionality and performance, while the **behavior** of a system is what the system does to implement its function and is described by a sequence of states.

The **service** delivered by a system is its behavior as it is perceived by its **users**. The part of the provider's system boundary where service delivery takes place is the provider's **service interface**. The part of the provider's total state that is perceivable at the service interface is its **external state**, while the remaining part is its **internal state**. The delivered service is a sequence of the provider's external states.

Correct service is delivered when the service implements the system function. On the contrary, a **service failure** or failure is an event that occurs when the delivered service deviates from correct service. A service fails either because it does not comply with the functional specification, or because this specification did not adequately describe the system function. A service failure means that at least one or more external states of the system deviate from the correct service. The deviation, also called an **error**, is the part of the total state of the system that may lead to its subsequent service failure. The adjudged or hypothesized cause of an error is called a **fault**, which can be internal or external. A **vulnerability** is an internal fault that enables an external fault to harm the system. An attack is considered a malicious external fault that tries to provoke a service failure. An **attack** is basically the occurrence of a threat that compromises an asset or system resource by exploiting a vulnerability in the IT system [218].

The different ways in which the deviation is manifested are the system's service **failure modes**. The service failure modes can be characterized based on next four viewpoints:

- the failure domain;
- the detectability of failures;
- the consistency of failures; and
- the consequences of failures on the environment.

Dependability of a system is the ability to avoid service failures that are more frequent and more severe than what is acceptable, and, as a result, it can justifiably be trusted. Dependability comprises the properties **availability** (readiness for correct service), **reliability** (continuity of correct service), **safety** (absence of catastrophic consequences on the user(s) and the environment), **integrity** (absence of improper system alterations), and **maintainability** (ability to undergo modifications and repairs). Dependability is equivalent to **trustworthiness**, which implies the assurance that a system will perform as expected.

Security traditionally comprises **confidentiality** (absence of unauthorized disclosure of information), integrity and availability properties. **Authentication** is defined as the process of reliable security identification of subjects or data by incorporating an identifier and its authenticator [123]. The authentication of entity or data is also commonly referred as a desired security property.

A **dependability or security failure** occurs when the given system suffers service failure modes more frequently or more severely than acceptable.

6.2 System Model

In this Section, the signature creation and signature verification environments that will be used to design the taxonomy are modeled. This abstract representation of the systems will permit us to devise the categories of attacks that may subvert the signing or verification processes.

6.2.1 Signature creation environment

The model of the environment used by the signer to generate a digital signature is shown in Figure 6.1. It corresponds to the model provided by CEN CWA 14170 [46], with some refinements further explained. Therefore, we will consider digital signatures generated by end users, who own and control the signing key and interact with the signing capabilities offered by the environment. The physical environment where the signing process takes place can be either under the signer's control and possession (e.g. personal computer, corporate laptop, mobile phone, personal digital assistant, etc.) or operated by a service provider not necessarily related to or under the control of the signer (e.g. any public place like a metro station, bank, etc.), but in any case accessible by him. It should be mentioned that there is no direct interface or communication channel between the signer and the signing key and the information to be signed. The signer must rely on the IT elements of the environment to produce a digital signature. In any case, it is assumed that the environment has been designed to permit the signer to securely and consciously create digital signatures on his behalf and on the intended information.

As mentioned by CEN, the model does not intend to specify the nature or distribution of the components. These aspects can only become more concrete in the context of a particular set of technologies that apply to the signature creation system.

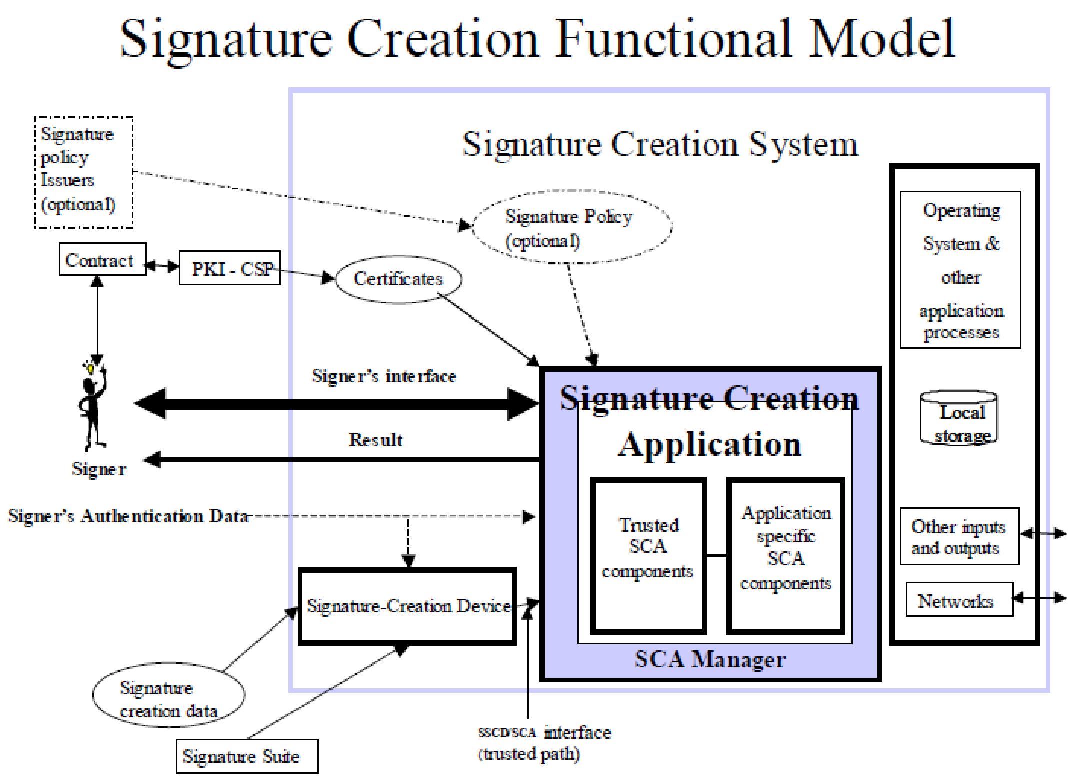
Signature Creation Functional Model
Signature Creation System
Signature policy Issuers (optional)
Contract
PKI - CSP
Certificates
Signature Policy (optional)
Operating System & other application processes
Local storage
Other inputs and outputs
Networks
Signer's interface
Result
Signer
Signer's Authentication Data
Signature Creation Application
Trusted SCA components
Application specific SCA components
SCA Manager
Signature-Creation Device
Signature creation data
Signature Suite
SSCD/SCA interface (trusted path)

Figure 6.1: The signature creation functional model (source [46]).

In the model of CEN, the Signature Creation Environment (SCE) is the physical, geographical and computational environment of the Signature Creation System (SCS), including the signer and the existent policies. The SCS consists of the software and hardware needed to generate digital signatures.

The Signature Creation Application (SCA) is the application within the SCS that creates digital signatures, excluding the Signature Creation Device (SCDev). According to the European Directive on electronic signatures [76], the SCDev can be either software (SW SCDev) or hardware (HW SCDev). A typical functional difference between software and hardware devices is that, in the former, the Signature Creation Data (SCD) are usually exportable, while in the latter they cannot be (theoretically) extracted from the device. On the other hand, CEN CWA 14169 [45] defines the security requirements for Secure Signature Creation Devices (SSCDev) in accordance with the Annex III of the European Directive, and following the technology-neutral principle claimed by it.

However, and contrary to this, CEN CWA 14170 actually restricts the attribution of SCDev and SSCDev to hardware devices only. We decided to follow the general approach given by the European Directive and CEN CWA 14169, also considering software devices as devices that implement the SCD. The reason also stems from the possibility that a digital signature not generated with a hardware cryptographic device can be considered as evidence in legal proceedings as well (see Chapter 2). Figure 6.2 shows a view of our adapted model of signature creation environment.

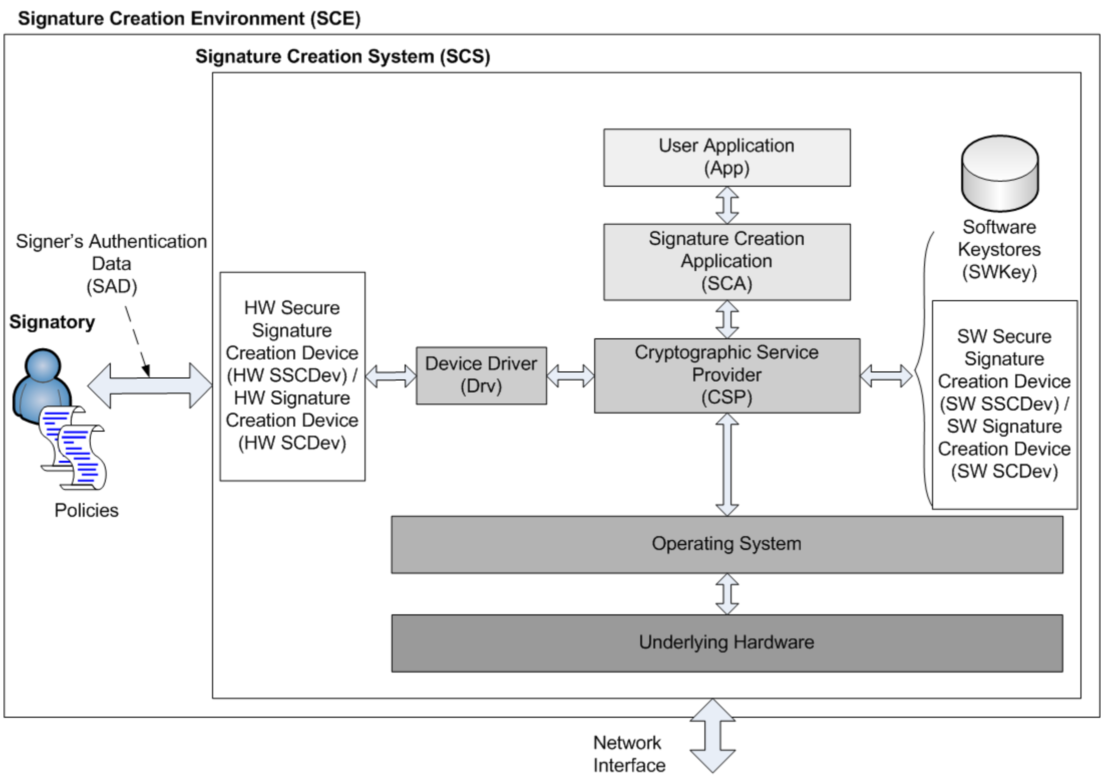
Signature Creation Environment (SCE)
Signature Creation System (SCS)
User Application (App)
Signature Creation Application (SCA)
Cryptographic Service Provider (CSP)
Software Keystores (SWKey)
SW Secure Signature Creation Device (SW SSCDev) / SW Signature Creation Device (SW SCDev)
HW Secure Signature Creation Device (HW SSCDev) / HW Signature Creation Device (HW SCDev)
Device Driver (Drv)
Signer's Authentication Data (SAD)
Signatory
Policies
Operating System
Underlying Hardware
Network Interface

Figure 6.2: Adapted signature creation functional model.

In addition to the software (S)SCDev, our adapted model shown in Figure 6.2 represents three additional components not found in CEN CWA 14170 model: the device driver, the cryptographic service provider and the software keystores. These elements are commonly found in a SCS, independently of its nature, and will permit us to discover relevant attack categories useful for the taxonomy.

The software keystores (SWKey) are protected data structures that store the SCD of the user(s), but do not implement signing capabilities. SWKey are managed by specific software, such as SCA or Web browsers. Access to the SCD stored in SSCDev, SCDev and SWKey is protected by means of the Signer's Authentication Data (SAD), which are the data (e.g. PIN, password or biometric data) used to authenticate the signer and required to allow the use of the SCD.

The signer can interact (service interface) with the SCA directly or through other applications, identified in Figure 6.2 as the User Application layer (App). The Cryptographic Service Provider (CSP) is a software layer that operates on top of the Operating System and that allows the SCA to transparently access and use the SCD by invoking the Application Programming Interface (API) that it publishes.

The Data To Be Signed (DTBS) - not shown in Figure 6.2 - is defined by [46] as the complete electronic data to be signed. It covers the signer's document (SD) and, optionally, the signature attributes, which enrich the semantic of the document. The SD can be a local document, a Web content, a document imported from a different environment or any other type of information. Signature attributes, if present, are signed together with the SD and may include, among others, the data content type (it expresses the encoding format of the SD), the signature policy reference or the commitment type made in the act of signing. Data To Be Signed Representation (DTBSR) is defined by [46] as the data sent by the SCA to the (S)SCDev for signing. DTBSR will generally correspond to the cryptographic hash of the DTBS.

6.2.2 Signature verification environment

CEN models the Signature Verification Environment (SVE) in CWA 14171 [47]. The model intends to outline a general guideline on signature verification procedures in order to achieve the recommendations for secure signature verification given in Annex IV of the European Directive on electronic signatures [76]. In a nutshell, the signature verification system is intended to permit the verifier to securely and unambiguously verify digital signatures and associated information.

CEN defines the verifier as *the entity which verifies the electronic signature*, and establishes that *it may be a single entity or multiple entities.* But contrary to CEN CWA 14170 (see Section 6.2.1), the verifier is not restricted to end users only. Although the European Directive [76] explicitly refers to the verifier as the person to whom to the data used for verifying the signature and the verification result are displayed, CEN considers three different models: a natural person, using his workstation and accompanying software to request verification of a received signature, a computer program, using an automated procedure, for which the term "display" would cover a broader meaning, and a third-party to which the verification could be sub-contracted.

In this Chapter we adhere to the vision given by the European Directive, and thus, assume that the verifier is a human user that physically visualizes the signed data and any other information that must be verified during the signature verification process. However, we comply with a multi-party verification process as long as there exists a participation of an end user. For example, the end user would be typically involved in the initial and subsequent verification (see [47]) in order to visualize and verify the signed data and signer's identity, while the validation information to be captured and archived during such stages may be leveraged to third parties. It should be noted that when referring to Signature Verification System (SVS) along the rest of the Chapter, we mean a system that may implement the initial verification, the subsequent verification, or both, by means of a Signature Verification Application (SVA).

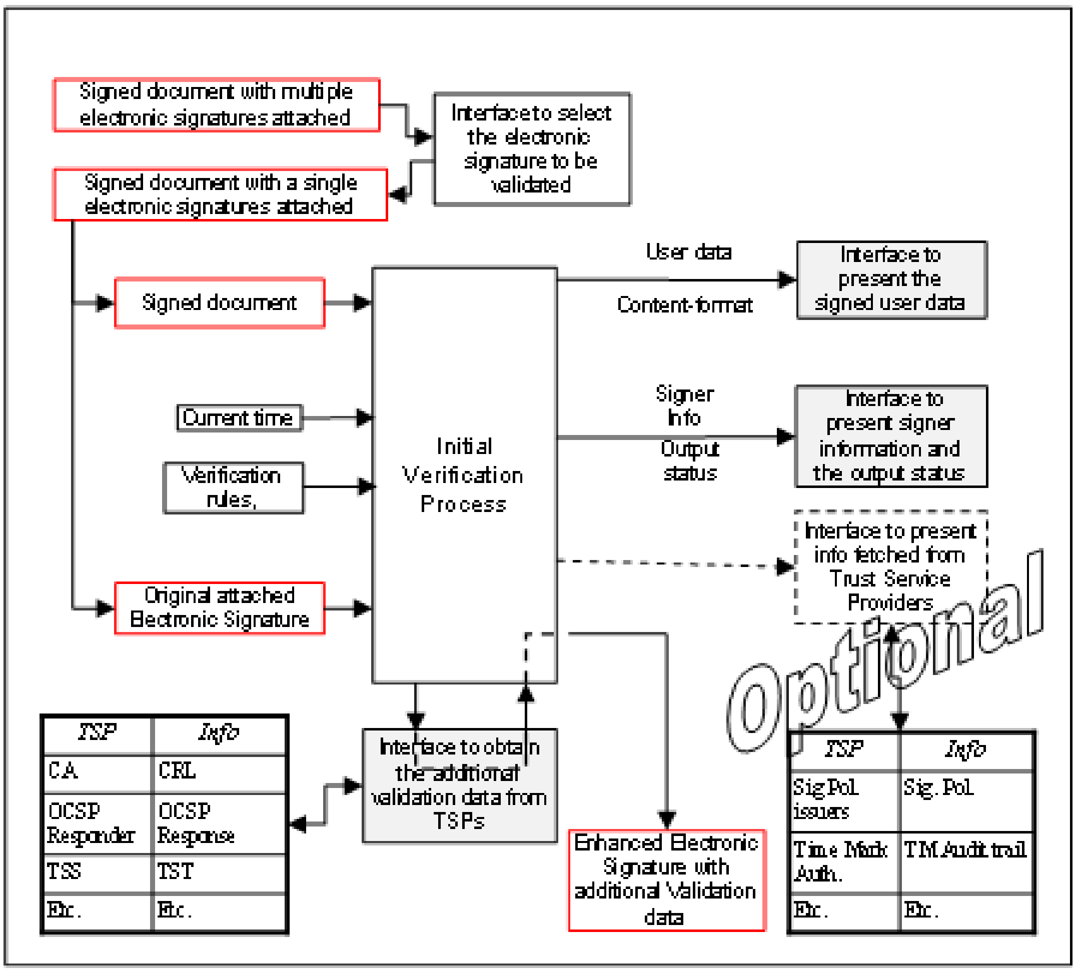
Signed document with multiple electronic signatures attached
Signed document with a single electronic signatures attached
Interface to select the electronic signature to be validated
Signed document
User data
Content-format
Interface to present the signed user data
Current time
Verification rules,
Initial Verification Process
Signer Info
Output status
Interface to present signer information and the output status
Interface to present info fetched from Trust Service Providers
Optional
Original attached Electronic Signature
TSP
Info
CA
CRL
OCSP Responder
OCSP Response
TSS
TST
Etc.
Etc.
Interface to obtain the additional validation data from TSPs
Enhanced Electronic Signature with additional Validation data
TSP
Info
Sig Pol issuers
Sig. Pol
Time Mark Auth.
TM Audit trail
Etc.
Etc.

Figure 6.3: The signature verification functional model - initial verification systems (source [47]).

The models of an initial verification system and a subsequent verification system defined by CEN are represented in Figures 6.3 and 6.4, respectively. While the initial verification stage could be partially performed by the signer, the subsequent verification stage is always performed by the verifier.

Along the rest of the Chapter we will use Data To Be Verified (DTBV) term to refer to the information that was signed and has to be verified against the signature. This information corresponds to both the signed document and the optional signed attributes, that is, the information contained in the DTBS during the signature generation. The DTBV would correspond to the signed document element represented in Figures 6.3 and 6.4.

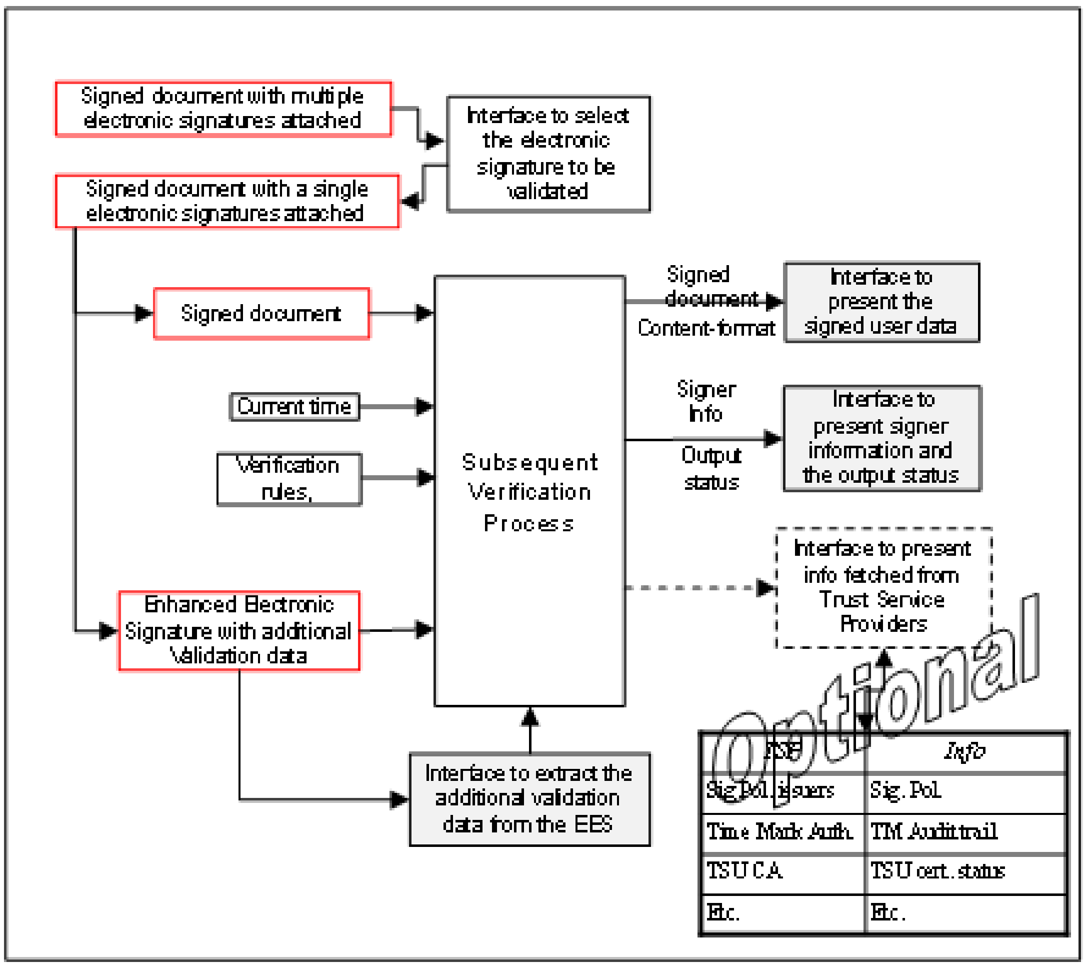
Signed document with multiple electronic signatures attached
Interface to select the electronic signature to be validated
Signed document with a single electronic signatures attached
Signed document
Current time
Verification rules,
Subsequent Verification Process
Signed document
Content-format
Interface to present the signed user data
Signer Info
Output status
Interface to present signer information and the output status
Interface to present info fetched from Trust Service Providers
Enhanced Electronic Signature with additional Validation data
Interface to extract the additional validation data from the EES
Optional
TSP
Info
Sig.Pol. issuers
Sig. Pol.
Time Mark Auth.
TM Audit trail
TSU CA
TSU cert. status
Etc.
Etc.

Figure 6.4: The signature verification functional model - subsequent verification systems (source [47]).

6.3 Threat Model

This Section provides an asset-centric threat model that will be later used to devise the taxonomy. The system model from which the threat model is built corresponds to the signature creation (SCE) and verification (SVE) environments described in Section 6.2. A system complying with this model could implement the signature creation functionality, the signature verification functionality, or both. Besides, specific implementations may not include some parts of the environments. In any case, the threat model is still applicable since the assets and security objectives remain the same independently of the particular products or technology used for the system development.

The next Section 6.3.1 defines the assets considered in the system model, and the security objectives and the security functional requirements to be fulfilled and implemented by it. The faults and service failures applicable to the system model are introduced in Section 6.3.2. The attacker profile is defined in Section 6.3.3. Finally, the assumptions on the threat model are given in Section 6.3.4.

6.3.1 Assets, security objectives and security functional requirements

The identification of the assets to be protected by the system is paramount during a threat model definition. It permits to further define the security objectives and requirements to be met.

Risk analysis and management methodologies usually consider a large number of assets, including logical, physical and personal entities, and that it would imply a violation of the purported dependability and security properties of the system if they were compromised by an attacker. We only consider the assets that affect the system function in the last term, that is, either permit the signer to securely and consciously create digital signatures on his behalf and on the intended information, or permit the verifier to securely and unambiguously verify digital signatures on the signed information.

For instance, there would be no signing or verification process reliable if an attacker compromised the integrity of the operating system. However, the attacker must compromise one of the assets identified herein to succeed. The mere compromise of the operating system does not imply that the security of a signing or verification process has been subverted. Therefore, we focus on the particular asset that, if compromised, might imply a system failure (see Section 6.3.2 for detailed information about considered systems' failures).

The assets considered in our threat model for the SCE are the next:

- The signer's authentication data (SAD);
- The signature creation data (SCD);
- The signer's document (SD);
- Additional signature attributes (Attr), like the data content type, commitment type made or signature policy reference;
- The set of explicit (automated) rules and policies to follow for the generation process (POL);
- Composition of the SD and the additional attributes (DTBS);
- The final representation (hash) of the information to be sent to the SCDev/SSCDev before the signature computation (DTBSR);
- The signature creation system (SCS), including executable and configuration files and any data needed by the SCS during its operation (e.g. data stored in volatile memory).

On the other hand, the assets considered for the SVE are:

- The signer's document (SD);
- The signer's certificate (SC);
- The electronic signature (ES) and signed attributes (Attr);
- The set of explicit (automated) rules and policies to follow for the verification process (POL);
- The signature verification system (SVS), including executable and configuration files and any data needed by the SVS during its operation (e.g. data stored in volatile memory).

For the purpose of the taxonomy defined in this Chapter, we will reduce the properties of interest that shape dependability and security concepts. The result is the security objectives to be achieved by the SCS and SVS. From the security perspective, we expect the SCS to achieve the next security objectives respecting the assets above:

- Confidentiality of the signer authentication data;
- Confidentiality of the signature creation data;
- Integrity of the signer's document including the syntax (prior signature computation) and semantic (post signature computation);
- Integrity of the signature attributes, the corresponding DTBS and DTBSR before it is sent to the SCDev/SSCDev;
- Integrity and authentication of the set of explicit (automated) rules and policies.

For the SVS, we expect the following security objectives achievement:

- Integrity and authentication of the signer's document (both syntactic and semantic as displayed to the verifier);
- Integrity and authentication of the signer's certificate (both syntactic and semantic as displayed to the verifier);
- Integrity of the electronic signature;
- Integrity and authentication of the signed attributes (both syntactic and semantic as displayed to the verifier);
- Integrity and authentication of the set of explicit (automated) rules and policies.

Therefore, we do not expect the SCS or SVS to assure a certain level of availability of the service. An attack that makes the signing service useless (e.g. the attacker modifies the signature creation data - violates its integrity) will prevent the signer from performing signatures. On the other hand, an attack that prevents the verifier from verifying a signature will provoke an incomplete verification. However, our concern are service failures further defined, which do not imply a degradation of the service availability but the undermining of the signature reliability.

From the dependability perspective, we expect the SCS and SVS to achieve the next security objectives:

- Integrity of the SCS itself (absence of improper system alterations).
- Integrity of the SVS itself (absence of improper system alterations).

Other properties inherent in dependability such as availability, reliability, safety and maintainability are not under the scope of our threat model, and thus, of our taxonomy.

Based on the security objectives defined above, we conclude that the high-level security functional requirements to be implemented by the system are:

- (SFR-1) The SCS must protect the signer from an incorrect or malicious use of the signature creation data.
- (SFR-2) The SCS must protect the signer from signing a document different than the intended one or under unintended conditions (i.e. rules and policies).
- (SFR-3) The SVS must protect the verifier from an ambiguous signature verification, presenting the actually signed information as intended by the signer.

The trace between assets, security objectives and security functional requirements to be fulfilled by the system is shown in Table 6.1. Each attack categorized in the taxonomy will necessarily violate one or more of these security objectives, and consequently provoke one of the three service failures defined in the next Section 6.3.2.

It was previously mentioned that only assets with an impact on the delivered service are being considered. In any case, it is important to remark that the security and dependability of external systems may impact on the SCS/SVS. Sometimes, there is a strong dependence between the system and the rest of entities that belong to the environment. As a result, the next security objectives should be met by the environment, though not reflected in the mapping of Table 6.1:

- Integrity of the rest of the entities of the environment upon which the system (SCS and/or SVS) depends;

Asset	Security objective	Security functional requirement
SAD	Confidentiality	SFR-1, SFR-2
SCD	Confidentiality	SFR-1, SFR-2
SD	Integrity	SFR-2, SFR-3
	Authentication	SFR-3
Attr	Integrity	SFR-2, SFR-3
	Authentication	SFR-3
DTBS	Integrity	SFR-2
DTBSR	Integrity	SFR-2
SC	Integrity	SFR-3
	Authentication	SFR-3
ES	Integrity	SFR-3
POL	Integrity	SFR-2, SFR-3
	Authentication	SFR-2, SFR-3
SCS	Integrity	SFR-1, SFR-2
SVS	Integrity	SFR-3

Table 6.1: Trace between assets, security objectives and security functional requirements

- Confidentiality of security attributes that may permit an attacker to gain access to system service (e.g. credentials to access the SCS);

- Availability and reliability of external security services upon which the system depends (e.g. security mechanisms implemented by the underlying operating system, and that assures the integrity of the SCS and confidentiality of internal channels; appropriate and timely update of the certificate revocation lists by the certification authority, etc.).

6.3.2 Faults and service failures

As previously defined, the service failure is an event that occurs when the delivered service deviates from the correct service. It implies an error in the system behavior that has been caused by one or more internal and/or external faults. In our model, the system function is to permit the signer to securely and consciously create digital signatures on his behalf and on the intended signer's document (for SCS), and/or to permit the verifier to securely and unambiguously verify digital signatures and the identity of the signer (for SVS). We assume that the specification does adequately describe the system function. Consequently, we consider a service failure as an occurrence of one of the next events:

- (1) The SCS does not protect the signer from an unintended or unauthorized use of the signature creation data.
- (2) The SCS does not protect the signer from signing a document different than the intended one or under unintended conditions (i.e. rules and policies).
- (3) The SVS does not protect the verifier from performing an ambiguous signature verification.

An attack on the SCS will try to provoke service failure (1) and/or (2), while an attack on the SVS will focus on service failure (3). In the system model discussed in Section 6.2, it was explained that the SCS and SVS depend on the underlying operating system, other applications running at user or kernel level, the physical environment and other aspects like the administration and security policies to be enforced in the environment. As a result, a failure of a service different than the one provided by the SCS or SVS may indirectly have an impact on it, and thus provoke a SCS or SVS service failure as well. Therefore, the attacker may try to provoke a service failure in the SCS or SVS by directly interacting with it or exploiting a vulnerability in the SCS or SVS (internal fault), or by provoking a failure in a different service upon which the SCS or SVS depends (external fault that propagates errors into the SCS/SVS by interaction or interference).

According to failure modes listed in Section 6.1, and defined in [17], an attack on the SCS/SVS could be successfully mounted if service failures (1), (2) or (3) manifested in a *failure domain* mode. Failure domain mode can be distinguished as a content failure (i.e. the content of the information delivered at the service interface deviates from implementing the system function) or timing failure (i.e. the time of arrival or the duration of the information delivered at the service interface deviates from implementing the system function). For example, if the signature generation or signature verification service fails, showing a different document to sign or verify (content failure), respectively, then the attacker could trick the user during the signing or verification process.

From the *detectability failure* mode viewpoint, an attack on the SCS/SVS will succeed if the attacker provokes an *unsignaled failure* (caused by a latent error), either because the service failure is not detected or because the service failure is not signaled at the service interface (it would not be noticeable by the end user) by the implemented detection mechanism. Otherwise, the signer or verifier is expected to perform the adequate corrective actions (e.g. the signer revokes the digital certificate that corresponds to the involved private key, avoiding the attacker to gain a benefit from the service failure). In should be mentioned that, though service failure (2) does not always imply that the signature creation data has been compromised, the signer should follow a conservative behavior.

The *consistency of failures* implies that the incorrect service is perceived by two or more users either identically (consistent failure) or differently (inconsistent failure). For the purpose of our taxonomy, this mode of failure is not relevant and thus it is not considered for the attack categories.

Last, the failure mode *consequences of failures on the system environment* can be graded with a severity level, and are normally associated with a maximum acceptable probability of occurrence. As discussed in this thesis, we consider that the current definition and legal consequences of digital signatures as non-repudiation evidence imply that the probability of an attacker to successfully obtain a fraudulent signature, or a fraudulent signature be verified as valid, must be reduced to the minimum (zero if it was technically possible). As a result, the failure severity for the three service failures described above should be set by the system manufacturer to the gravest one. In the context of our taxonomy, service failures (1), (2) and (3) identified above are always dependability or security failures as their occurrence is, by definition, more severe than what the service should accept (a successful attack has occurred).

6.3.3 Attacker profile

We consider two properties to profile the attackers: the attack potential and the capability to access or approach the target of the attack.

The attack potential is defined as the perceived likelihood of success should an attack be launched, expressed in terms of the attacker's ability (i.e. expertise and resources) and motivation [218]. We consider an attacker with enough expertise, resources and motivation to execute any potential attack, provided that it is technically feasible.

Respecting access capabilities, we consider attackers that can carry out both internal and external attacks.

In an internal attack, the attacker operates inside the security perimeter of the environment, and can be either (i) a malware that has infected an IT element of the system, (ii) a physical person that directly interacts with the environment, handles the hardware (e.g. the (S)SCDev) or even communicates with the end user, or (iii) the end user itself (i.e. a malicious signer). Regarding attacks that handle the hardware (ii), we consider attackers that can perform invasive tampering attacks on the hardware (e.g. micro probing techniques). This type of attacks needs to handle the hardware and physically harm it. On the other hand, the advantage of non-invasive attacks is that the equipment used in the attack can usually be disguised as a normal device (e.g. smartcard reader), and thus the owner of the compromised hardware might not notice that the secret keys have been stolen. Therefore it is unlikely that the validity of the compromised keys will be revoked before they are abused [140].

On the contrary, in an external attack, the attacker operates outside the security perimeter of the signature environment, possibly through the network.

As can be seen in Figure 6.2, the signer is included by CEN in the SCE model, as suggested by safety engineering best practices [205]. However, CEN does not include the entity that represents the verifier in the SVE model. We also consider that it is not important for the taxonomy, as the verifier does not possess any secret, and thus the attacker cannot obtain any benefit from him.

6.3.4 Assumptions on the threat model

An important issue when defining a threat model is to establish the assumptions made on the model. In particular, the next two assumptions are made.

- The environment is untrusted. The SCE and SVE are operated by end users, and sometimes owned by them. As such, no assumption on security policies, trustworthiness of the underlying platform and so forth should be made. Any mistaken decision could be made by the user when maintaining and operating the environment, who normally has no technical knowledge. Due to the unavoidable presence or occurrence of faults, systems are never completely trustworthy. Design flaws and code bugs in underlying software or hardware are an endless source of vulnerabilities that an attacker can exploit to break the security of digital signature processes. According to a survey made by OCDE in 2008, 43% of Internet users from United States have suffered some sort of malware infection in their home PC [160]. Another recent study carried out in 2009 revealed that approximately the 33% of computers protected with an updated anti-virus are infected, while the percentage of those not protected increased to 46% [198]. On the other hand, APWG scanned 22 millions of computers, from which 48,35% were infected [185]. In the 2009 annual report, Panda indicated that, in some countries, the percentage of infected computers reached more than 60% [10].

- For the purpose of the present taxonomy, third parties (e.g. certification authority, validation authority, time-stamping authority, etc.) are considered trustworthy. Consequently, the taxonomy will not cover categories that represent the execution of attacks on these entities.

6.4 A Taxonomy of Attacks on Digital Signatures

This Section presents the taxonomy, which is based on the next dimensions. Each category is assigned an identifier that consists of the number of dimension it belongs to (D) and a category or subcategory number (CAT) according to the hierarchical order established:

- **Attacker's goal**, which covers the goal of the attack.
- **Method of attack**, which corresponds to the method of attack executed by the attacker to achieve the goal classified in the previous dimension.
- **Target of the attack**, which identifies the target(s) of the attack. The multiple instances of this dimension permit to know every element, both software and hardware, that is affected during the attack.

6.4.1 Dimension one: Attacker's goal

The goal of an attack will consist of achieving one of the service failures (1), (2) or (3) described in Section 6.3.2. In particular, there are six categories in this dimension, with no further refinement:

D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions

The attacker does not directly use the signature creation data (SCD) but seeks to deceive the signer to unconsciously sign a document that is of benefit to the attacker, against the signer's interests, or both. The attacker may also modify the rules or policies that establish the signing requirements, possibly weakening them for his further benefit. This category corresponds to service failure (2).

D1-CAT2: Unauthorized use of the signature creation data (SCD)

The attacker seeks to use the SCD on behalf of the user, but without his consent and knowledge. For that purpose, the attacker will need to either obtain the signature creation data or have access to the signing function at will. This category corresponds to service failure (1).

D1-CAT3: Replace signed information

The attacker seeks to directly replace part of or the whole signed information for his own benefit, the signer's detriment or both, and once the signature has been computed.

This category corresponds to service failure (2), in the sense that the final signed document does not correspond to the original one.

D1-CAT4: Make the signed document be attributed to a user different than the actual signer

The attacker seeks that a document signed by certain signer is verified as signed by a different entity. Thereby, the attacker could provoke a wrong document's authorship attribution. For instance, the attacker may seek that a document signed by another one is verified as signed by himself (e.g. the document's content is beneficial). The attacker may also seek that a document not signed by a certain user is verified as signed by the user (e.g. the document's content is detrimental to the user). This category corresponds to service failure (3).

D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content

The attacker seeks that the signed document and/or signed attributes are shown to the verifier either with a content which appearance may vary (polymorphic) or with a content different to what was actually signed or was intended to be signed. For instance, if the attacker is the signer, he may seek to make some content that has not signed in the beginning be verified as such (e.g. for his own benefit). If the attacker is an external malicious entity, he may seek to attribute to the signer some content not signed or intended to be signed by the signer (e.g. to damage the signer's interests). This category corresponds to service failure (3).

There is an exception when the attacker seeks to show a different content with regard to the identity of the signer (e.g. target the signed attribute signing-certificate, specified in CAdES [74] and XAdES formats [75]). In this case, goal *D1-CAT4: Make the signed document be attributed to a user different than the actual signer* prevails, and thus the attack should be classified accordingly.

D1-CAT6: Make the signature validity verification conclude with an opposite result

The attacker seeks to make a signature validity verification raise a result different than the correct one. The validity of the signature depends not only on the signature itself but also on the certificate validity. This goal covers both when a valid signature is verified as invalid, and when an invalid signature is verified as valid. For example, if the attacker is the signer, he may seek that a signature signed by himself is verified as invalid (e.g. to repudiate the commitment made in the signed document), while if the attacker is an external malicious entity, he may seek to make a valid signature generated by a certain user be verified as invalid (e.g. to damage the signer's interests). In the opposite direction, the attacker may seek to make a signature generated over a fraudulently modified document be verified as valid. This category corresponds to service failure (3).

6.4.2 Dimension two: Method of attack

The methods that can be used by the attacker to achieve the identified goal are specified in this dimension, and illustrated in Figures 6.5 and 6.6. Seven categories have been devised at the first level, which are further refined into subsequent subcategories:

D2-CAT1: Environment manipulation

This category includes the methods aimed at manipulating the environment of the SCS/SVS in order to have an effect on the signature creation process or the signed information once the signature has been computed.

D2-CAT2: Modification prior to signature computation

This category contains the attack methods that take part before the signature computation, and which goal is the modification of the information to be signed, either directly (modification of the data to be signed) or indirectly (fraudulent data are included by reference from the data to be signed).

D2-CAT2.1: Document modification. This subcategory of methods relates to modifications performed in the document to be signed.

D2-CAT2.1.1: Dynamic content inclusion. This subcategory of methods implies the inclusion of dynamic content into the document to be signed. These methods aim at maintaining the document's integrity while varying its semantic.

D2-CAT2.1.1.1: Hidden code. The attacker inserts special tags or fields in the document to be signed. These hidden code will be translated into certain value depending on specific conditions that can be controlled by the attacker.

D2-CAT2.1.1.2: Active code. The attacker inserts special code, like scripts or macros, in the document to be signed. This code is executed during the signed document opening or visualization, and thus can perform several operations like changing the content being shown.

D2-CAT2.1.1.3: Linked content. The attacker inserts links in the document to be signed that point to external content not controlled by the signer. Once the signature is performed, the attacker can manipulate that external content at will.

D2-CAT2.1.2: Content modification. The attacker modifies the content of the document to be signed, but without including any sort of dynamic content (e. g. modification of the text of the document to be signed).

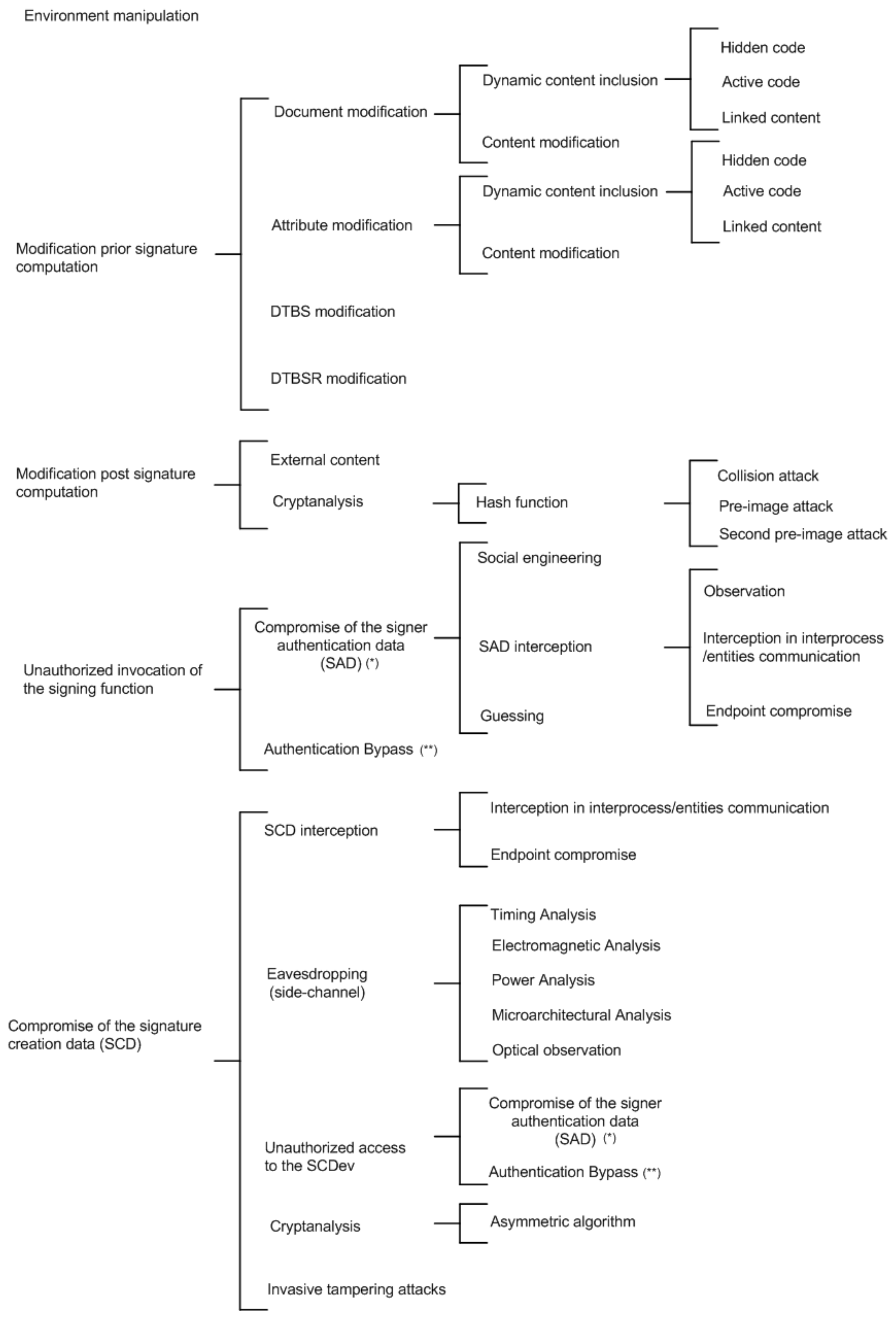
Environment manipulation
Modification prior signature computation
Document modification
Dynamic content inclusion
Hidden code
Active code
Linked content
Content modification
Attribute modification
Dynamic content inclusion
Hidden code
Active code
Linked content
Content modification
DTBS modification
DTBSR modification
Modification post signature computation
External content
Cryptanalysis
Hash function
Collision attack
Pre-image attack
Second pre-image attack
Unauthorized invocation of the signing function
Compromise of the signer authentication data (SAD) (*)
Social engineering
SAD interception
Observation
Interception in interprocess /entities communication
Endpoint compromise
Guessing
Authentication Bypass (**)
Compromise of the signature creation data (SCD)
SCD interception
Interception in interprocess/entities communication
Endpoint compromise
Eavesdropping (side-channel)
Timing Analysis
Electromagnetic Analysis
Power Analysis
Microarchitectural Analysis
Optical observation
Unauthorized access to the SCDev
Compromise of the signer authentication data (SAD) (*)
Authentication Bypass (**)
Cryptanalysis
Asymmetric algorithm
Invasive tampering attacks

Figure 6.5: Dimension "method of attack" (first 5 categories).

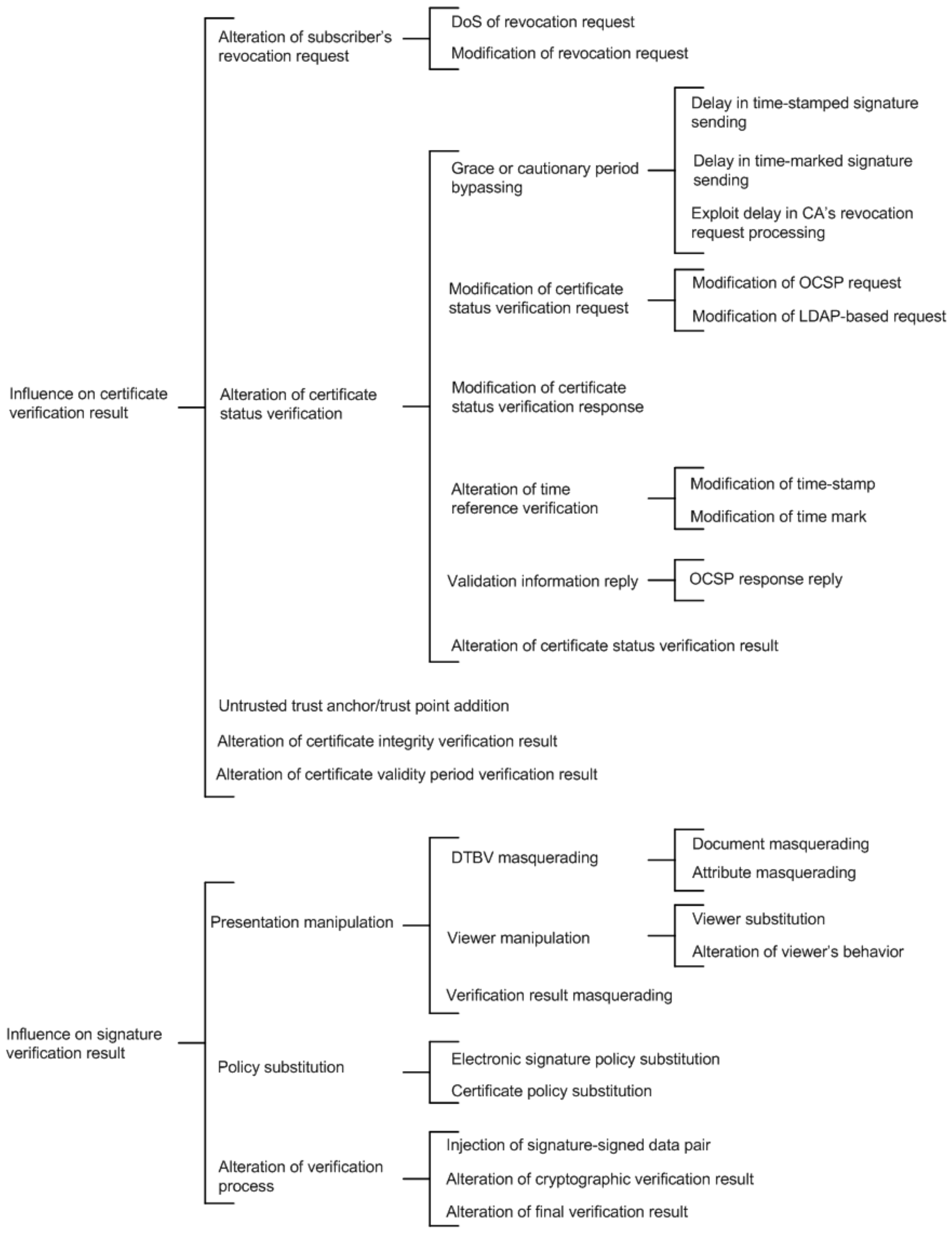
Influence on certificate verification result
Alteration of subscriber's revocation request
DoS of revocation request
Modification of revocation request
Alteration of certificate status verification
Grace or cautionary period bypassing
Delay in time-stamped signature sending
Delay in time-marked signature sending
Exploit delay in CA's revocation request processing
Modification of certificate status verification request
Modification of OCSP request
Modification of LDAP-based request
Modification of certificate status verification response
Alteration of time reference verification
Modification of time-stamp
Modification of time mark
Validation information reply
OCSP response reply
Alteration of certificate status verification result
Untrusted trust anchor/trust point addition
Alteration of certificate integrity verification result
Alteration of certificate validity period verification result
Influence on signature verification result
Presentation manipulation
DTBV masquerading
Document masquerading
Attribute masquerading
Viewer manipulation
Viewer substitution
Alteration of viewer's behavior
Verification result masquerading
Policy substitution
Electronic signature policy substitution
Certificate policy substitution
Alteration of verification process
Injection of signature-signed data pair
Alteration of cryptographic verification result
Alteration of final verification result

Figure 6.6: Dimension "method of attack" (last 2 categories).

D2-CAT2.2: Attribute modification. This subcategory of methods relates to modifications performed in the attributes to be signed.

D2-CAT2.2.1: Dynamic content inclusion. This subcategory of methods implies the inclusion of dynamic content into the attributes to be signed, aiming at maintaining the attributes' integrity while varying their semantic.

D2-CAT2.2.1.1: Hidden code. The attacker inserts special tags or fields in the attributes to be signed. These hidden code will be translated into certain value depending on specific conditions that can be controlled by the attacker.

D2-CAT2.2.1.2: Active code. The attacker inserts special code, like scripts or macros, in the attributes to be signed. This code is executed during the attributes enforcement or visualization, and thus can perform several operations like changing the content being shown.

D2-CAT2.2.1.3: Linked content. The attacker inserts links in the attributes to be signed that point to external content not controlled by the signer. Once the signature is performed, the attacker can manipulate that external content at will.

D2-CAT2.2.2: Content modification. The attacker modifies the content of the attributes to be signed, but without including any sort of dynamic content.

D2-CAT2.3: DTBS modification. The attacker modifies the information that represents the data to be signed.

D2-CAT2.4: DTBSR modification. The attacker modifies the hash of the data to be signed. This would be the last data transformation step before the signature is computed.

D2-CAT3: Modification post signature computation

This category contains the methods that take part once the signature has been computed, and which goal is the modification of the signed information, either signed directly (modification of the signed data) or indirectly (modification of data referenced from the signed data).

D2-CAT3.1: External content. The attacker modifies information referenced from the signed information (e.g. XSD, DTD). The difference between this method and *D2-CAT2.1.1.3: Linked content* or *D2-CAT2.2.1.3: Linked content* lies in that, in the former, the link to the external content is not included by the attacker, while in the latter, the link is explicitly inserted by the attacker.

D2-CAT3.2: Cryptanalysis. The attacker applies a cryptanalytic method to generate a document different than the signed one without breaking the signature validity.

D2-CAT3.2.1: Hash function. The attacker applies methods specifically focused on breaking the security of the hash function used in the signature computation. Assuming a hash function that generates a n-bit output, there are three possible attacks.

D2-CAT3.2.1.1: Collision attack. The attacker is able to find a pair of messages $M \neq M'$ where $hash(M) = hash(M')$ with a complexity lower than $O(2^{n/2})$ (e. g. The birthday attack).

D2-CAT3.2.1.2: Preimage attack. The attacker, given a hash value H, is able to find a message M' where $H = hash(M')$ with a complexity lower that $O(2^n)$.

D2-CAT3.2.1.3: Second preimage attack. The attacker, given one message M, is able to find a second message, M', $M' \neq M$ to satisfy $hash(M) = hash(M')$ with a complexity lower than $O(2^n)$

D2-CAT4: Unauthorized invocation of the signing function

This category collects the methods that do not permit the attacker to know the SCD but to make use of it without the user's consent and knowledge.

D2-CAT4.1: Compromise of the signer authentication data (SAD). This subcategory covers the methods that permit the attacker to retrieve the SAD.

D2-CAT4.1.1: Social engineering. The attacker manipulates or tricks the signer to reveal the SAD.

D2-CAT4.1.2: SAD interception. The attacker intercepts the SAD during the SCS operation.

D2-CAT4.1.2.1: Observation. The attacker observes the SAD while the signer enters it in the SCS (i.e. shoulder surfing).

D2-CAT4.1.2.2: Interception in interprocess/entities communication. The attacker intercepts the SAD during its transmission between logical or physical processes or entities belonging to the SCS (e.g. sniffing techniques, software keyloggers, hooks,...).

D2-CAT4.1.2.3: Endpoint compromise. By having compromised a process or entity belonging to the SCS, and that intervene during the communication of SAD inside the SCS, the attacker is able to intercept the SAD when used (i.e. hardware keyloggers).

D2-CAT4.1.3: Guessing. The attacker uses a probabilistic method, brute force or keyboard acoustic emanation techniques to guess the SAD.

D2-CAT4.2: Authentication Bypass. The attacker bypasses the authentication method. As a result, the attacker is able to invoke the signing function without even knowing the SAD.

D2-CAT5: Compromise of the signature creation data (SCD)

This category includes the methods that permit the attacker to retrieve the SCD. Attacks classified under this category are the most dangerous ones, since the attacker would be able to make use of the SCD at will, even in a different environment.

D2-CAT5.1: SCD interception. The attacker intercepts the SCD during the creation or issuance processes.

D2-CAT5.1.1: Interception in interprocess/entities communication. The attacker intercepts the SCD during its transmission between logical or physical processes or entities.

D2-CAT5.1.2: Endpoint compromise. By having compromised a process or entity involved in the SCD creation, issuance, management or operation within the SCE boundaries, the attacker is able to retrieve the SCD.

D2-CAT5.2: Eavesdropping (side-channel). Side-channel attacks exploit the information leakage from physical characteristics of the hardware during the execution of the cryptographic algorithm. Thereby, the cryptographic key can be guessed, and thus compromised. It does not matter the complexity or security of the mathematical algorithm, because the fundamentals of side-channel attacks rely on the dependencies between the data processed (e.g. the private key) and/or the operation performed by the cryptographic device (e.g. smart card) and the physical behavior of the underlying hardware.

D2-CAT5.2.1: Timing Analysis. A Timing Analysis attack exploits timing measurements from vulnerable systems to find the entire secret keys.

D2-CAT5.2.2: Electromagnetic Analysis. An Electromagnetic Analysis attack exploits correlations between secret data and variations in power radiations emitted by tamper-resistant devices, like smart cards.

D2-CAT5.2.3: Power Analysis. A Power Analysis attack analyses the relationship between the power consumption of a cryptographic device and the handled data during cryptographic operations.

D2-CAT5.2.4: Microarchitectural Analysis. Microarchitectural Analysis (MA) studies the effects of common processor components and their functionalities on the security of software cryptosystems. MA attacks exploit the microarchitectural components of a processor to obtain the cryptographic keys. These attacks are purely based on software, and can compromise the security system despite of the implemented security techniques, such as virtualization, sandboxing or memory protection.

D2-CAT5.2.5: Optical observation. Optical emanations can leak sensitive information. If the information being processed corresponds to the SCD, the attacker may compromise it by simply observing the optical signal being produced.

D2-CAT5.3: Unauthorized access to the SCDev. The attacker compromises the SCD by accessing the (S)SCDev (or software keystore) where it is stored.

D2-CAT5.3.1: Compromise of the signer authentication data (SAD). The attacker is able to retrieve the SCD once the SAD is known. This method requires the SCD to be exportable. This subcategory is further refined using the same subcategories as *D2-CAT4.1 Compromise of the signer authentication data (SAD).*

D2-CAT5.3.2: Authentication Bypass. The attacker is able to access the SCD even without knowing the SAD. This method requires the SCD to be readable by an entity different than the SCDev or the software keystore.

D2-CAT5.4: Cryptanalysis. The attacker applies a cryptanalytic method to discover the signature creation data.

D2-CAT5.4.1: Asymmetric algorithm. This subcategory collects attacks focused on obtaining the private key used in an asymmetric algorithm. There are several asymmetric or public key algorithms (i.e. RSA, DSA, Elliptic Curve, etc.). Depending on the algorithm, the set of possible attack methods varies.

D2-CAT5.5: Invasive tampering attacks. In these attacks, the hardware (secure) signature creation device is physically tampered using special equipment. This subcategory would collect attacks that retrieve the signature creation data using decapsulation and micro probing techniques, advanced beam technologies, etc.

D2-CAT6: Influence on certificate verification result

This category includes methods of attack that have an impact during the verification of the certificate associated to the signature being verified. Some methods can be used to make a verifier conclude that either an invalid certificate is valid or that a valid certificate is invalid.

D2-CAT6.1: Alteration of subscriber's revocation request. The attacker alters the request made by the subscriber (legitimate owner of the certificate and associated private key) to revoke the certificate. This method of attack is oriented to avoid the revocation of such certificate.

D2-CAT6.1.1: DoS of revocation request. The attacker performs a denial of service (DoS) attack by preventing the request from reaching the certification authority in charge of processing the revocation.

D2-CAT6.1.2: Modification of revocation request. The attacker modifies the information of the request that identifies the certificate which revocation is being requested.

D2-CAT6.2: Alteration of certificate status verification. The attacker alters the certificate status verification process, making the verifier to conclude that an invalid certificate (i.e. revoked or suspended) is valid, or that a valid certificate is invalid.

D2-CAT6.2.1: Grace or cautionary period bypassing. This subcategory collects methods that allow the attacker to bypass or make the grace/cautionary period, as defined in [47], ineffective.

D2-CAT6.2.1.1: Delay in time-stamped signature sending. The attacker delays the time-stamped signature sending until the CRL is updated. This method of attack assumes that the legitimate owner of the certificate (user) cannot detect the private key compromise before the attacker makes use of the signed document and the corresponding signature. To implement this method, the attacker must have compromised the private key, generated a signature on behalf of the user and time-stamped it on his own. As a result, when the verifier receives the signature, he will possess a CRL issued after the signing time (specified by the time-stamp) and thus will not wait for any further update, considering the signature as valid.

D2-CAT6.2.1.2: Delay in time-marked signature sending. This method of attack is similar to *D2-CAT6.2.1.1: Delay in time-stamped signature sending*, with the difference that a time-mark is used instead of a time-stamp.

D2-CAT6.2.1.3: Exploit delay in CA's revocation request processing. This method of attack exploits the inevitable time that a certification authority needs to update the CRL since the revocation request is received and processed. Therefore, and assuming that the attacker can use the private key of the victim, the attacker is able to enforce a signed document even though the owner had requested the revocation of the associated certificate.

D2-CAT6.2.2: Modification of certificate status verification request. The attacker alters the certificate status request made by the verifier in order to prevent him from discovering the actual revocation status, or query the status of a revoked certificate different than the purported one.

D2-CAT6.2.2.1: Modification of OCSP request. The attacker modifies the field *serialNumber* of the OCSP request structure [173]. This method makes the verifier to request the status of a certificate different than the targeted one. In case the OCSP request is to be signed by the requester, then the attacker should perform the modification before the signing or compromise the OCSP signing key for a further modification and signature calculation. Also, as the standard establishes that the response must include the certificate serial number (to ascertain that the response is given for the desired certificate), and for this attack to succeed, the attacker should launch a secondary attack of type *D2-CAT6.2.3: Modification of certificate status verification response*, modifying the certificate serial number of the response.

D2-CAT6.2.2.2: Modification of LDAP-based request. This method of attack is similar to *D2-CAT6.2.2.1: Modification of OCSP request*, but being applied over a Lightweight Directory Access Protocol (LDAP) request.

D2-CAT6.2.3: Modification of certificate status verification response. This subcategory represents methods that intend to modify the certificate status response given by the authority (e.g. certification authority, OCSP responder, etc.). The modification should be performed in a manner that the verifier accepts the message as valid and authentic.

D2-CAT6.2.4: Alteration of time reference verification. The attacker modifies the token used by the verifier as the time reference. The methods represented herein imply that the modification is made in a manner that it cannot be detected by the verifier.

D2-CAT6.2.4.1: Modification of time-stamp. The attacker modifies the time reference included in the signature time-stamp in order to prevent the verifier from detecting the actual revocation status of the certificate at the time when the signature was generated.

D2-CAT6.2.4.2: Modification of time mark. The attacker modifies the time reference included in the signature time-mark in order to prevent the verifier from detecting the actual revocation status of the certificate at the time when the signature was generated. For this attack to be executed, the attacker needs to intercept the time-mark information sent by the time-mark authority to the verifier.

D2-CAT6.2.5: Validation information reply. The attacker re-uses validation information in order to prevent the verifier from detecting the actual revocation status of the certificate.

D2-CAT6.2.5.1: OCSP response reply. The attacker replies with an outdated OCSP response that contains the *certStatus* field [173] set to a value of interest to the attacker (e.g. 'good', for a certificate that is currently revoked, or 'unknown'/'revoked' for a certificate that is currently valid). Due to the time verification requirements established in Section 4.2.2.1 of standard [57], the attacker should modify the current time of the verifier's machine for this attack to succeed.

D2-CAT6.2.6: Alteration of certificate status verification result. The attacker intercepts the routine that performs the status verification process and alters the result that indicates the status of the certificate.

D2-CAT6.3: Untrusted trust anchor/trust point addition. The attacker injects a new trust anchor or trust point to make a certificate owned by the attacker be considered as trusted by the verifier during the certification chain verification. The attacker poses as the victim by using a certificate containing as subject Distinguished Name (DN) the victim's DN.

D2-CAT6.4: Alteration of certificate integrity verification result. The attacker intercepts the routine that performs the certificate integrity verification process and alters the result.

D2-CAT6.5: Alteration of certificate validity period verification result. This subcategory includes methods where the attacker alters the verification of the validity period of the certificate. For instance, the attacker may intercept the routine that performs the validity period verification process and alter the result to make the certificate be regarded as valid. The attacker may also modify the current time of the verifier's machine.

D2-CAT7: Influence on signature verification result

This category includes methods of attack that affect the verification of the signature being verified. Some methods can be used to make a verifier conclude that either an invalid signature is valid or that a valid signature is invalid. Methods specifically focused on influencing the verification of the signing certificate are included in *D2-CAT6: Influence on certificate verification result* category.

D2-CAT7.1: Presentation manipulation. This subcategory collects methods that manipulate the way the Data To Be Verified (DTBV) are visualized by the verifier. This set of methods violates the What-Is-Presented-Is-What-Is-Signed (WIPIWIS) principle.

D2-CAT7.1.1: DTBV masquerading. The attacker alters the visualization of the DTBV, being able to present a DTBV different than what has been actually signed. This subcategory represents methods that focus on the way the DTBV is shown to the verifier, but independently of the viewer being used (e.g. superimposing text on the signed document during its visualization).

D2-CAT7.1.1.1: Document masquerading. The attacker alters the visualization of the signed document.

D2-CAT7.1.1.2: Attribute masquerading. The attacker alters the visualization of one or more signed attributes.

D2-CAT7.1.2: Viewer manipulation. The attacker manipulates the viewer used to present the DTBV. In this case, the methods lie in achieving a different visualized DTBV by targeting the viewer used.

D2-CAT7.1.2.1: Viewer substitution. The attacker substitutes the viewer with another one that presents the DTBV in a different manner.

D2-CAT7.1.2.2: Alteration of viewer's behavior. The attacker alters the behavior of the viewer to make it present the DTBV in a different manner.

D2-CAT7.1.3: Verification result masquerading. The attacker manipulates the signature verification result shown to the verifier. A valid signature may be presented as invalid, or an invalid signature as valid.

D2-CAT7.2: Policy substitution. The attacker replaces a policy used for the signature verification.

D2-CAT7.2.1: Electronic signature policy substitution. The attacker replaces the electronic signature policy referenced in the signature and that contains the clauses and requirements that establish the conditions under which the signature should be considered as valid.

D2-CAT7.2.2: Certificate policy substitution. The attacker replaces the certificate policy referenced in the certificate and that contains a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements.

D2-CAT7.3: Alteration of verification process. This subcategory includes methods that affect the processes that implement the signature verification process, in a manner that the achieved result differs from what is expected.

D2-CAT7.3.1: Injection of signature-signed data pair. The attacker replaces the information during the verification process by injecting a pair of signed document-signature. It is assumed that the attacker possesses a document signed by the signer and the corresponding signature, but different to the signed document and signature that is to be verified.

D2-CAT7.3.2: Alteration of cryptographic verification result. The attacker intercepts the routine that performs the cryptographic verification process and alters the result.

D2-CAT7.3.3: Alteration of final verification result. The attacker is able to alter the final result of the signature verification process, influencing on the conclusion about the validity or invalidity of the signature.

Not every method of attack permits the attacker to achieve every attacker's goal established for the first dimension. Table 6.2 relates the categories of the first dimension with the categories in the first level of the second dimension.

6.4.3 Dimension three: Target of the attack

This dimension classifies the target(s) of the attack. An attack can target more than one victim at the same time, resulting in multiple entries in this dimension. It does not mean that the mutual exclusion principle is violated (see Section 10.1). A non-mutually exclusive taxonomy would produce two different entries for the same target. In this case, several targets may need to be classified for the same attack. A scheme of the categories and subcategories of this dimension is depicted in figure 6.7.

Next, the categories and subcategories of this dimension are listed:

D3-CAT1: Cryptography

D3-CAT2: Software

D3-CAT2.1: Application.

D3-CAT2.1.1: External application.

D3-CAT2.1.1.1: Kernel level application.

D3-CAT2.1.1.2: User level application.

D3-CAT2.1.2: Related application.

D3-CAT2.1.2.1: Document processor.

D3-CAT2.1.2.2: SCA.

D3-CAT2.1.2.3: CSP.

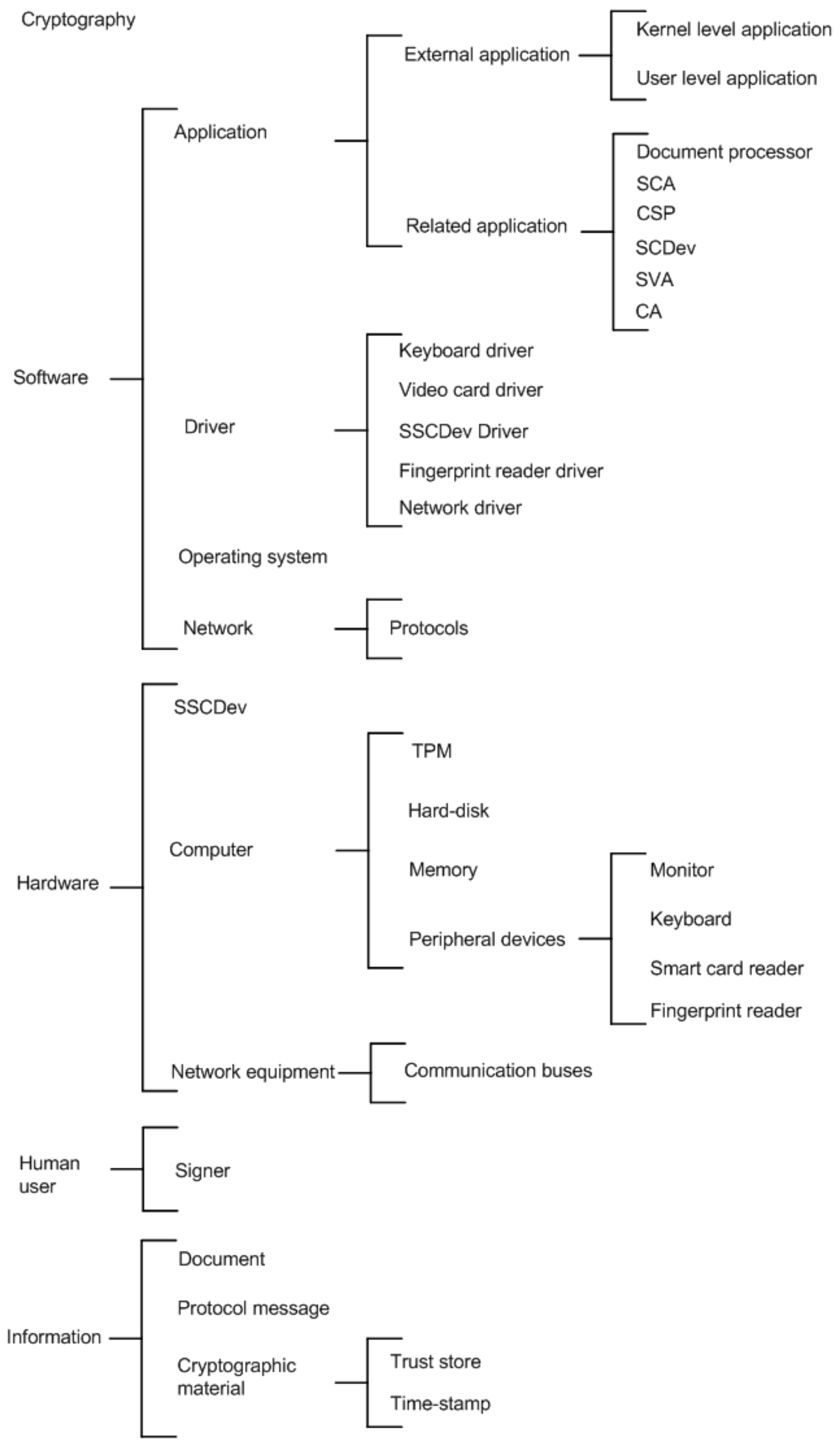

Figure 6.7: Dimension "target of the attack".

Goal	**Method**
D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions	D2-CAT1: Environment manipulation D2-CAT2: Modification prior to signature computation
D1-CAT2: Unauthorized use of the signature creation data (SCD)	D2-CAT4: Unauthorized invocation of the signing function D2-CAT5: Compromise of the signature creation data (SCD)
D1-CAT3: Replace signed information	D2-CAT3: Modification post signature computation
D1-CAT4: Make the signed document be attributed to a user different than the actual signer	D2-CAT6: Influence on certificate verification result D2-CAT7: Influence on signature verification result
D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content	D2-CAT1: Environment manipulation D2-CAT7: Influence on signature verification result
D1-CAT6: Make the signature validity verification conclude with an opposite result	D2-CAT6: Influence on certificate verification result D2-CAT7: Influence on signature verification result

Table 6.2: Relationship between the Dimension *Attacker's Goal* and Dimension *Method of Attack*

D3-CAT2.1.2.4: SCDev[1].

D3-CAT2.1.2.5: SVA.

D3-CAT2.1.2.6: CA.

D3-CAT2.2: Driver.

D3-CAT2.2.1: Keyboard driver.

D3-CAT2.2.2: Video card driver.

D3-CAT2.2.3: SSCDev driver.

D3-CAT2.2.4: Fingerprint reader driver.

D3-CAT2.2.5: Network driver.

D3-CAT2.3: Operating system.

D3-CAT2.4: Network.

D3-CAT2.4.1: Protocols.

D3-CAT3: Hardware

[1]This subcategory includes the software signature creation device and the software keystore as defined in Section 6.2.1

D3-CAT3.1: SSCDev.
D3-CAT3.2: Computer.
D3-CAT3.2.1: TPM (Trusted Platform Module).
D3-CAT3.2.2: Hard-disk.
D3-CAT3.2.3: Memory.
D3-CAT3.2.4: Peripheral devices.
D3-CAT3.2.4.1: Monitor.
D3-CAT3.2.4.2: Keyboard.
D3-CAT3.2.4.3: Smart card reader.
D3-CAT3.2.4.4: Fingerprint reader.
D3-CAT3.3: Network equipment.
D3-CAT3.3.1: Communication buses.

D3-CAT4: Human user

D3-CAT4.1: Signer.

D3-CAT5: Information

D3-CAT5.1: Document.
D3-CAT5.2: Protocol message.
D3-CAT5.3: Cryptographic material.
D3-CAT5.3.1: Trust store.
D3-CAT5.3.2: Time-stamp.

6.5 Method of Classification

The method of classification associated to a taxonomy must clearly guide a user when a new element has to be classified. The next steps should be followed when classifying an attack under our taxonomy:

1. **Attack analysis**. The attack must be analyzed in order to understand its behavior and features. Depending on the available information, the result of the analysis will be more or less detailed and accurate. This information should, at least, permit the completion of the remaining steps of the classification method.

2. **Identification and classification of the attacker's goal**. The goal of the attack must be identified and classified according to the dimension *Attacker's Goal*.

3. **Analysis and classification of the method of attack**. The method used by the attacker to achieve the identified goal must be classified according to the dimension *Method of Attack* and Table 6.2. The method of attack must be classified in a subcategory of the deepest level of the selected branch.

4. **Identification and classification of targets of the attack**. The elements affected by the attack must be identified and classified in accordance with the subcategories found in dimension *Target of the Attack*. Like in the previous step, a subcategory of the deepest level of the selected branch must be selected. Only targets directly involved in the signature creation or verification operation should be classified. For instance, any internal attack carried out by means of malware must firstly compromise the system (e.g. due to a vulnerability in the operating system). However, the operating system should not be classified as a target unless it had a vulnerability that allowed the attacker to directly compromise the generation/verification process.

5. **Refine the taxonomy**. In steps 2, 3 and 4, if a more specific or refined subcategory is needed, it must be added to the taxonomy, and the attack classified accordingly.

As a result, an attack will be classified using one category of dimension *Attacker's Goal*, one subcategory of dimension *Method of Attack*, and one or more subcategories of dimension *Target of the Attack*.

As mentioned above, the accuracy and detail extracted from the attack analysis depends on the available information. Obscure attacks or attacks from which little information can be obtained will necessarily be more complicated to classify. On the other hand, attacks that can be studied in detail, for instance applying re-engineering techniques to the malware code or during a forensic study, will provide much more information that can be used to accurately define the attack behavior and features, and thus to classify the attack.

Next, and for illustration purposes, a Trojan horse attack on software for electronic signatures [219] is classified using our method of classification.

Firstly, we have studied the information provided in [219]. The attack is carried out on two of the most deployed signature software in Germany. The attacker obtains a handle to the PIN edit control in a Windows operating system environment. Once the user has entered the PIN, the attacker is able to retrieve it and start as many signing processes as desired. The authors do not provide the attack vector to infect the system with the Trojan horse, though we can assume that the environment does not provide effective protection for detecting this specific malware. Otherwise, the attack would have been thwarted in the very beginning. Therefore, it can be assumed that there is a vulnerability in the system that the attacker can exploit for the infection.

In the second step, and using the information above, the goal of the attack has to be classified. In this case, the main objective of the attacker is to use the signature creation data without the user's consent. Therefore, we classify the goal as *D1-CAT2: Unauthorized use of the signature creation data (SCD).*

The next step establishes that the method used by the attacker to achieve the identified goal must be classified. Table 6.2 restricts the candidates to two. Because the attack does not retrieve the SCD, and due to the description given, it is obvious that the method intends to use the SCD by invoking the signing function. Therefore, we classify the method of attack as *D2-CAT4: Unauthorized invocation of the signing function*. However, the method of classification stipulates that a subcategory of the deepest level of the selected branch must be selected. Taking a look at D2-CAT4 subcategories, it is clear that the attacker is compromising the signer's authentication data (*D2-CAT4.1: Compromise of the signer authentication data (SAD)*). In particular, the attacker obtains the PIN. Then, in a deeper classification, the next subcategory corresponds to *D2-CAT4.1.2: SAD interception*, and more specifically, *D2-CAT4.1.2.3: Endpoint compromise*, as the PIN is retrieved due to a vulnerability in the PIN edit control of the signature creation application.

Finally, the description of the attack permits us to identify the target of the attack, which is the SCA.

The next Table shows the final result of the classification.

Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.1: Compromise of the signer authentication data (SAD) → D2-CAT4.1.2: SAD interception → D2-CAT4.1.2.3: Endpoint compromise
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA

It is important to highlight that every method is subject to the subjectivity of the user in charge of the classification. Even when the available information of the attack is very detailed, two different users can reach contradictory conclusions. The training, skills and perspective of the user are paramount to make the correct decision. Even sometimes there is not only a single correct decision, but many.

For example, the same attack can be correctly classified in two different manners depending on the viewpoint taken. An attack that injects dynamic content into the document to be signed can be classified according to the goal dimension as *D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions*, if the attacker is not the signer, and the user concludes that the process being subverted is the signature generation, or as *D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content*, if the attacker is the signer itself or a different malicious entity, but the user considers that the process being subverted is the signature verification. Possibly, the goal pursed by the attacker is both of them, that is, both deceiving the signer respecting the information being signed and deceiving the verifier respecting the information signed.

The method proposed herein intends to reduce the ambiguity during the classification procedure, but we do not claim that the method is deterministic, since we consider that it is not possible in this inexact field of study.

6.6 Chapter Summary

In this Chapter we have proposed the first holistic taxonomy of attacks on digital signatures, including attacks on the generation and verification phases. This taxonomy will help developers to build digital signature technology more robust and resilient to current threats, as the knowledge has been categorized in a manner that permits devising general countermeasures independently of the particular technology or implementation being used.

The context of the taxonomy has been defined following a threat modeling approach, delimiting the boundaries, restrictions and assumptions under which this taxonomy might be useful and applicable. In addition, a method of classification has been provided to permit the systematic classification of an attack on digital signatures according to the taxonomy.

Chapter 7

Division of the Signature Environment

In this Chapter we present a new paradigm to effectively enhance the reliability of digital signature-based evidence. Firstly, Section 7.1 provides an overview of our novel approach. The specific security mechanisms that permit its implementation and other practical issues are given in Section 7.2. Finally, we conclude the Chapter in Section 7.3.

7.1 Overview

In this Chapter we evolve the split trust paradigm analyzed in Chapter 5 to a more rigorous model in order to substantially reduce the probability of an attack that may subvert the reliability of digital signature-based evidence. Current split trust paradigm shifts all sensitive operations to a single trusted environment. As the analysis of the state of the art demonstrates that no environment can be sufficiently trusted, we propose to divide the evidence generation and verification processes into several environments, all of them untrusted. In a nutshell, we apply the famous Caesar's quotation *Divide et vinces*. As a result, by increasing the number of environments needed in conjunction to generate and verify the evidence, the reliability of the evidence itself is enhanced.

In case of evidence generation, each environment generates one or more digital signatures that are part of non-repudiation evidence. Evidence cannot be enforced as such until every signature is generated. Consequently, evidence is not based on a single signature anymore but multiple signatures. In case of evidence verification, the verification process is considered as reliable in case every environment, or a minimum number established as the threshold, reaches the same conclusion about the evidence validity. In both cases, more than one environment is mandatory to complete the generation or verification processes. Obviously, there must be a trade-off between the added complexity and the security improvement.

However, how do we know that certain signature has been generated from a particular environment, and as a result the division principle has been enforced? A digital signature is generated by the signer using his signature creation data (SCD). But, in general, a SCD can be indistinctly used at any environment. As a result, a mechanism to bind certain SCD with a particular environment is absolutely necessary. If the SCD is bound with a particular environment E, then it can be ascertained that any digital signature generated with that SCD has been generated at environment E. This Chapter deals with this necessity, and proposes some solutions to fulfill it.

From here on, *environment* corresponds to a signature environment (e.g. a PC, a mobile device, etc.), being either a signature creation environment, a signature verification environment, or both. *Evidence subject* is the end user the generates the digital signature-based evidence, while *evidence user* is the end user or end users that verify the digital signature-based evidence. Following definitions also apply for the remainder of the Chapter:

Definition 1. We define *evidence establishment* as an action by which either the evidence subject creates the evidence or the evidence user verifies the evidence. We consider evidence that consists of digital signatures. Therefore, evidence establishment implies either a signing operation on certain data or a digital signature verification operation.

Definition 2. An *attack on an environment* is an attack carried out by a malicious agent (active intruder or resident malware) which purpose is to obtain some benefit from the evidence establishment capabilities of that environment. Potential attacks that may be carried out by the malicious agent include those defined in the taxonomy of attacks of Chapter 6.

Definition 3. The *probability of a successful attack (PSA)* on an environment depends on both the probability of a malicious agent (attacker) to gain access to that environment (undermine the environment's security measures) and the probability of that attacker to subvert the specific security measures implemented by the environment to protect the evidence establishment capabilities.

7.2 Implementing the Division Principle

The signature environment division principle has several practical implications, as non-repudiation evidence is now based on multiple digital signatures. In this Section a guide for implementing the division principle is given, including the security mechanisms to generate and verify multi-signature based evidence.

7.2.1 General rules

During **evidence generation**, evidence subject must use more than one environment to generate the digital signatures that are part of the non-repudiation evidence. Each environment can be used to generate one or more signatures. The degree of reliability in the generation process depends on the number of environments used to produce the resultant evidence, and the PSA of each one (if known). Due to evidence nature and meaning (see Chapter 3), the evidence subject must be the same along the different environments used, as evidence is used to establish proof about an event or action, which traces back to a single origin.

During **evidence verification**, evidence user must use more than one environment to verify the evidence. Evidence may be partially verified, if it has not been completed yet, or completely verified, otherwise. The verification (partial or complete) must be performed in each environment. The degree of confidence in the actual validity of evidence depends upon the number of environments used, the PSA of each one (if known) and the results obtained in each verification. Depending on the transaction context, it can be established a minimum number of coherent verification results (threshold) or oblige that every verification reaches the same conclusion. More than one evidence user may participate in a collaborative manner during evidence verification.

The security mechanism that permits to implement a multi-signature based evidence consists of using parallel, sequential and/or embedded electronic signatures:

- Parallel signatures are applied on the same piece of information. They are mutually independent signatures where the order of the signatures is not important.
- Sequential signatures differ from parallel signatures in that the order is significant.
- Embedded signatures imply that one signature is applied to another. The sequence in which the signatures are applied is important and there is a strong interrelationship.

Current standardized electronic signature formats support the generation of any of these types of signature, including the generation of complex trees of signatures where parallel, sequential and embedded signatures may co-exist.

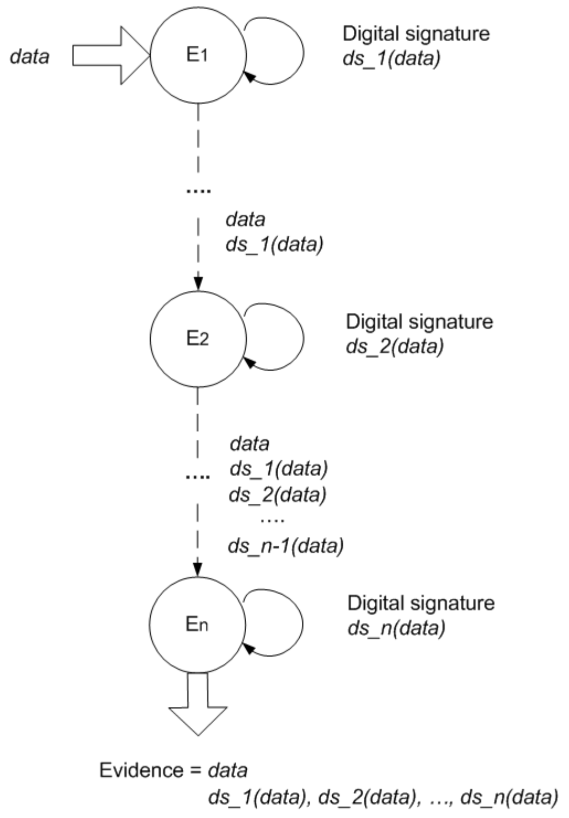

Figure 7.1: Chaining mode evidence generation scheme using sequential signatures.

7.2.2 Security mechanisms for evidence generation

This Section details the security mechanisms that can be used to generate a multi-signature based evidence following the division principle. Evidence subject must be the same along every environment used.

7.2.2.1 Chaining mode

In the chaining mode, the generation of the digital signatures which the evidence consists of must follow a certain order respecting the environments being used. Figure 7.1 represents the chaining mode scheme to generate evidence that consists of multiple sequential signatures.

As can be seen in the Figure, evidence subject starts the chain of signatures at environment E_1, by generating the first signature $ds\_1(data)$ over some data. The way the data is generated or obtained is irrelevant, provided that the signing process is carried out within E_1 boundaries. Subsequently, evidence subject transmits the *data* along with $ds\_1(data)$ to the second environment E_2, where the second signature $ds\_2(data)$ is generated, and so forth until the last environment E_n produces the last signature $ds\_n(data)$. Again, the specific means used to transmit such information between environments are irrelevant. The evidence subject may have physically transmitted the information by using a portable storage device or may have sent it through the network. The evidence subject may also receive such information from a different entity, due to a certain network or application protocol in which both participate. In any case, the order in which the environments are used, and thus the sequential signatures are generated, is paramount to generate valid evidence.

In the scheme shown in Figure 7.1, every digital signature corresponds to a sequential signature, as all of them are applied on the same piece of information (*data*). Parallel signatures cannot be used in this scheme as they are, by definition, order-independent signatures. However, a second scheme where embedded signatures are used is valid as well, as depicted in Figure 7.2. Due to the strong interrelationship between embedded signatures, the order followed for evidence generation is also fundamental to produce valid evidence.

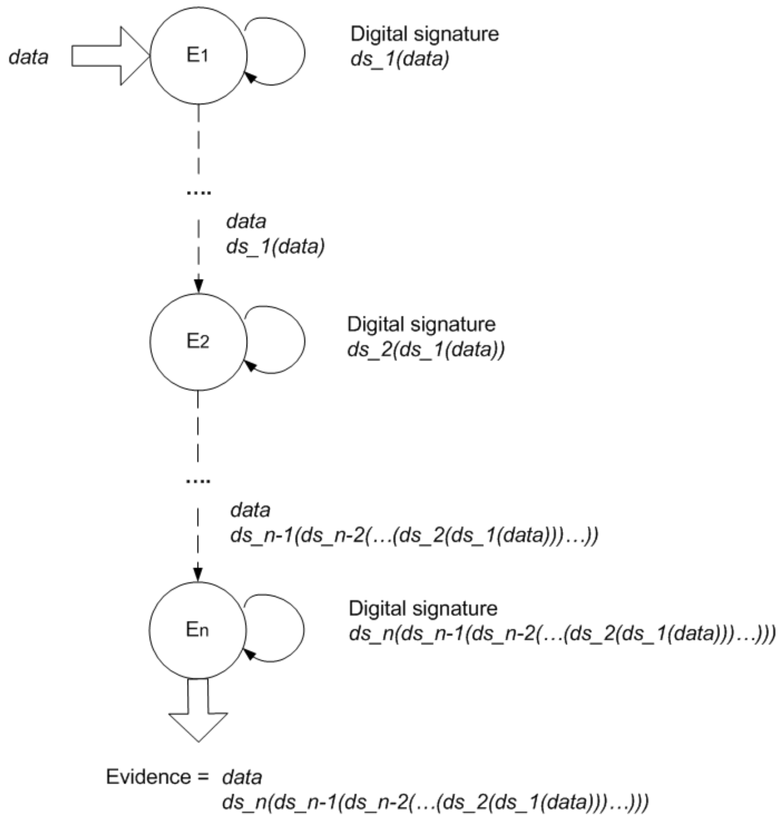

Figure 7.2: Chaining mode evidence generation scheme using embedded signatures.

7.2.2.2 Independent mode

In the independent mode, the evidence generation process does not need to follow a certain order respecting the environments being used. Next Figure 7.3 represents the independent mode scheme to generate evidence that consists of multiple parallel signatures. This type of signature is recommended for this scheme mode, as parallel signatures are mutually independent signatures where their order is not important.

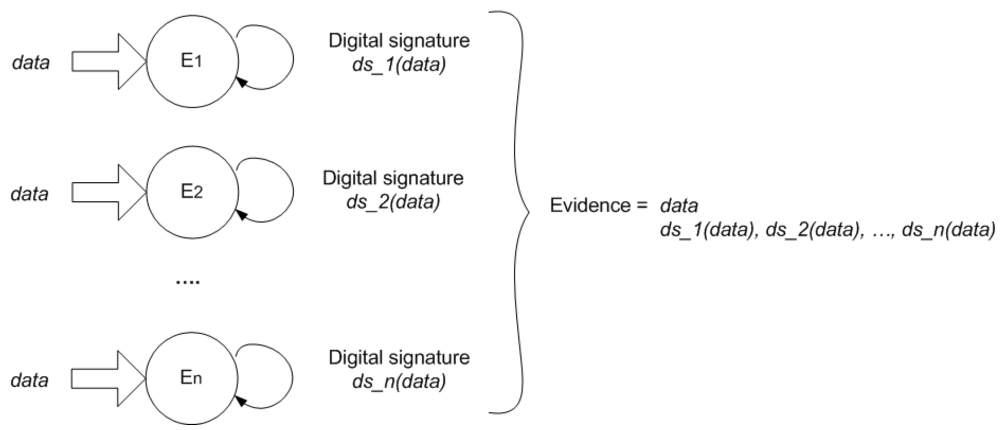

Figure 7.3: Independent mode evidence generation scheme using parallel signatures.

As shown in the Figure, each environment receives the same piece of information (*data*) on which the corresponding signature is generated. The resultant evidence will consist of all the signatures generated by every environment, along with the signed data.

7.2.3 Security mechanisms for evidence verification

This Section details the security mechanisms that can be used to verify a multi-signature based evidence following the division principle. It should be noted that there could be more than one evidence user along the different environment used and that collaborate during the evidence verification.

7.2.3.1 Chaining mode

In the chaining mode, and like during evidence generation, the evidence verification process must follow a certain order respecting the environments being used. Next Figure 7.4 depicts the evidence verification scheme when a chaining mode is applied. Evidence can be based on parallel, sequential or embedded signatures. Like in evidence generation following the chaining mode, the means used to transmit the evidence between the environments is not important, provided that the order is maintained.

It should be mentioned that evidence verification can be performed either once evidence is complete or during an evidence generation procedure, to ascertain that evidence is being generated properly. In the latter, and if evidence is being generated following a chaining mode scheme, as shown in Figures 7.1 and 7.2, its content changes from intermediate states, where the first and subsequent signatures are incorporated, to the final state, where the complete evidence is consolidated with the last signature.

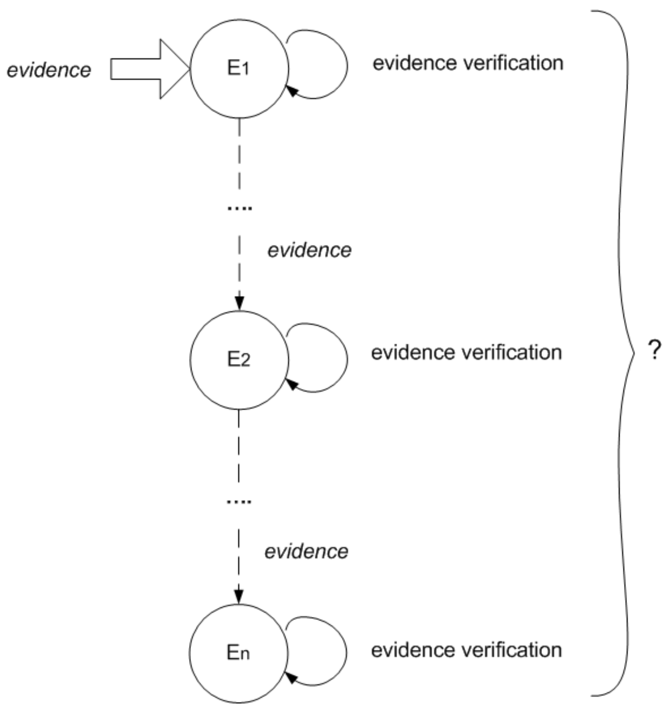

Figure 7.4: Chaining mode evidence verification scheme.

Therefore, during partial evidence verification each environment will verify the evidence in a different state. In any case, the accumulative nature of the chaining mode in the evidence generation process makes that, in a further environment, previous signatures are always verified again along with the new ones. Also, and contrary to the evidence generation case, there can be more than one evidence user that shared their validation results to achieve a collaborative evidence verification.

In order to consider the evidence as valid, the minimum number of environments at which the result of the verification must be satisfactory has to be specified. This number is called the threshold, and can vary from at least 2 to the total number of environments used. This condition should be defined in the non-repudiation policy in effect, and is formally defined as follows:

$$\exists\ N\ \subset\ M\ /\ \forall\ x\ \in\ N,\ x_i\ = 1,\ i\ =\ 0...n \tag{7.1}$$

Where M is the set of environments used to verify the evidence, and n is the threshold. If n is equal to the size of set M, then every environment must conclude with a satisfactory evidence verification (represented as $x_i\ = 1$) in order to consider the evidence as valid.

The threshold condition must be dynamically adapted depending on the verification stage, that is, a partial evidence verification or the complete evidence verification. For instance, in case of using 5 environments to verify the evidence, and during the partial evidence verification at the third environment, the threshold can have, at maximum, the value of 3.

7.2.3.2 Independent mode

In the independent mode, the evidence verification process does not need to follow a certain order respecting the environments being used. Next Figure 7.5 shows the evidence verification scheme in case an independent mode is used. Like in a chaining mode evidence verification, evidence can be based on parallel, sequential or embedded signatures.

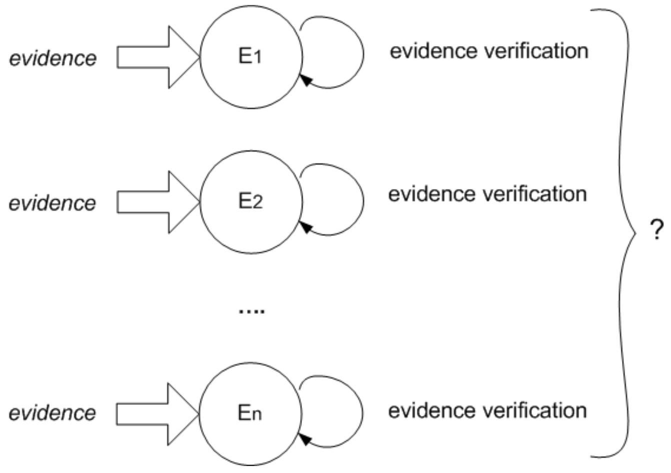

Figure 7.5: Independent mode evidence verification scheme.

In this scheme, evidence being verified will normally correspond to a complete evidence, though it could happen that different environments verify partial evidence in different states, like in the chaining mode. Likewise, there can be several evidence users that share their validation results to achieve a collaborative evidence verification. The threshold condition must also be specified like in Section 7.2.3.1.

7.2.4 Combination of schemes

Any combination of the schemes explained in Sections 7.2.2.1 and 7.2.2.2 is possible. Therefore, a particular scheme where parallel, sequential and embedded signatures coexist to produce non-repudiation evidence can be implemented. The evidence subject will have to take into account the order of the environments when required (i.e. sequential and embedded signatures).

On the other hand, as evidence verification implies that every signature that is part of evidence (in its current state) is verified - against data in case of parallel or sequential signatures, or against another signature in case of embedded signatures -, then the number and type of the signatures are transparent to the type of evidence verification scheme being used.

7.2.5 The binding between the environment and the signature creation data

Previous Chapter 6 identified a set of assets to protect, as they are the principal elements involved in a digital signature process. All of them, except the signature creation system and signature verification system, are generally independent of the environment. That is, they can be created or managed in any environment with digital signature capabilities. For instance, the signature creation data (SCD) can be stored in a software signature creation device or a hardware signature creation device, like a smart card. In both cases, the signature creation device (SCDev), and thus the SCD it contains, can be potentially used or imported into any environment.

Every environment is subject to suffer security threats with a given probability. As explained above, the division principle relies on the lower probability achieved when several environments are used in conjunction to produce non-repudiation evidence. The problem lies in that the reliability enhancement is achieved in this proposal thanks to the usage of several environments, but each required digital signature is generated by using certain cryptographic data (i.e. SCD) not initially bound with any environment.

Current standards on digital signatures do not address how to record information of the environment where a digital signature has been generated. We could define a particular signed attribute for an advanced electronic signature format where a unique identifier of the environment was included. But this information is easily forged by an attacker once the identifier is known. In other circumstances, the environment can be inferred, like, for example, if a mobile device with a cryptographic card is used. The SCDev (and thus the SCD) and the environment are physically linked to each other. But, in many other situations, the linkage between the SCD and the environment cannot be established.

In general, the SCD can be indistinctly used at any environment (e.g. when using smart cards or SCD stored in software). If the user used the same SCD in several environments, it would give an attacker that subverted the security of one or few of those environments the chance to compromise or access the SCD, or invoke the signing functionality when the SCD is available (see Chapter 6 for a detailed description of possible goals pursued by the attacker). Therefore, the attacker would be capable of generating all signatures that are required for producing the valid evidence. Moreover, in case every SCD was retrieved (attack method *D2-CAT5: Compromise of the signature creation data (SCD)*), the attacker could even use his own environment to compose the evidence. In conclusion, the lack of binding between the environment and the SCD makes the division of the signature environment useless.

Consequently, it is absolutely necessary to uniquely bind each SCD with a given environment, in a manner that evidence produced completely depends on the environments established to implement the division principle. We propose two possibilities to resolve this issue, and which are explained next.

7.2.5.1 By procedure

In the first approach, the binding between a SCD and an environment is purely procedural. The user is obliged by the non-repudiation policy in force to use a different SCD in each environment. As no technical mechanism is implemented to enforce the binding, the guarantee relies on the correct behavior of the user. However, the non-repudiation policy, if established by a competent authority, can include a statement by which any user that adheres to the policy (probably signing a contract with the aforementioned authority) and that is suspected to have failed to fulfill the requirements established in it (i.e. not using a different SCD in each environment) will be bound to the non-repudiation evidence generated in a given transaction where the policy applies. This measure would prevent users from obtaining a benefit in case they claimed, with no further proof, that they followed the binding by procedure requirement when they actually did not.

In a binding by procedure approach, there would be one or more SCD bound with each environment. In any case, each SCD must not be shared between different environments. Therefore, in a scenario of n environments, the user must own, at least, n different SCD.

Though an attacker compromised $n-1$ environments, and thus $n-1$ SCD, he would not be able to produce the valid evidence as he would lack of the n^{th} SCD. For that purpose, the user must assure that one SCD is not used out of the boundaries of the associated environment. Otherwise, the division principle is broken. As a single SCDev can store several SCDs, each SCD of these n SCDs must be stored in a different SCDev, avoiding that the same SCD is available at different environments when using the SCDev.

For instance, let's consider the next example, where there are three environments and the user owns three different SCDevs, with a different SCD each:

Environment E_1
Environment E_2
Environment E_3
User U with SCD SCD_{U_1}
User U with SCD SCD_{U_2}
User U with SCD SCD_{U_3}

In order to enforce the division principle, the next bindings between the user's SCDs and the environments are imposed:

E_1 with SCD_{U_1}
E_2 with SCD_{U_2}
E_3 with SCD_{U_3}

In case of a chaining mode scheme (see Section 7.2.2.1) (a) and an independent mode scheme (see Section 7.2.2.2) (b), valid evidence consists of:

Evidence = data, $S^{E_3}_{SCD_{U_3}}\left(S^{E_2}_{SCD_{U_2}}\left(S^{E_1}_{SCD_{U_1}}(data)\right)\right)$ (a)

Evidence = data, $S^{E_1}_{SCD_{U_1}}(data)$, $S^{E_2}_{SCD_{U_2}}(data)$, $S^{E_3}_{SCD_{U_3}}(data)$ (b)

7.2.5.2 By environment attestation

Another solution consists of making the environment attest to the signatures that are generated within its boundaries. That is, the environment would possess its own SCD, which would be used to "certify" those signatures, by, for instance, countersigning them. There should be an association of user SCD with environment SCD in a manner that if an environment attests to a signature generated with a user's SCD not bound with, then the resultant attested signature is not valid. In this case, the number of user's SCDs could be lower, as the attestation signature from the environment is mandatory.

For instance, let's consider the next example, where there are three environments and the user owns a single SCDev-SCD:

Environment E_1 with SCD SCD_{E_1}
Environment E_2 with SCD SCD_{E_2}
Environment E_3 with SCD SCD_{E_3}
User U with SCD SCD_{U_1}

In order to enforce the division principle, the next bindings between the user's SCD and the environments are imposed:

SCD_{E_1} with SCD_{U_1}
SCD_{E_2} with SCD_{U_1}
SCD_{E_3} with SCD_{U_1}

In case of a chaining mode scheme (see Section 7.2.2.1) (a) and an independent mode scheme (see Section 7.2.2.2) (b), valid evidence consists of:

Evidence = data, $S_{SCD_{E_3}}\left(S^{E_3}_{SCD_{U_1}}\left(S_{SCD_{E_2}}\left(S^{E_2}_{SCD_{U_1}}\left(S_{SCD_{E_1}}\left(S^{E_1}_{SCD_{U_1}}(data)\right)\right)\right)\right)\right)$
(a)
Evidence = data, $S_{SCD_{E_1}}\left(S^{E_1}_{SCD_{U_1}}(data)\right)$, $S_{SCD_{E_2}}\left(S^{E_2}_{SCD_{U_1}}(data)\right)$,
$S_{SCD_{E_3}}\left(S^{E_3}_{SCD_{U_1}}(data)\right)$ (b)

As can be seen, the user only needs to own a single SCDev-SCD. It is the environment attestation technique what guarantees the implementation of the division principle. If the user owned more than one SCD and made a mistake and unconsciously used certain SCD in an environment not bound with, the resultant evidence would not be valid.

As a result, a solution based on environment attestation assures that an attacker that compromises the user's SCDs but uses them out of the corresponding environment will not obtain any benefit from it. To subvert this solution, an attacker must:

- Either compromise every SCD of the user and the SCD of each environment, and produce the valid evidence accordingly using any desired environment (e.g. possibly his own one).
- Or compromise every SCD of the user but use each one from the corresponding associated environment.

The result is that, in both cases, the attacker still needs to compromise every associated environment to produce valid evidence. In the first case, obtaining the environments' SCD. In the second case, producing the evidence from those environments, what also implies compromising their security.

The environment attestation can be put into practice using the Trusted Computing Module (TPM) [226] or the Mobile Trusted Module (MTM) [227].

7.2.6 Some remarks

Some important remarks about implementation issues are given next.

7.2.6.1 Conscious data verification

In the schemes explained above, the evidence subject must be sure that the data over which the digital signatures are being generated is the desired one and the same along every environment (e.g. an attacker has not modified the data during transit). Otherwise, parallel and sequential digital signatures may still be cryptographically valid but evidence regarding the data could become inconsistent (different signatures generated over different data). To avoid such situation, and as mentioned above, partial evidence verification is recommended during its generation.

Because no environment is trusted, we cannot assume the existence of a trusted path from one environment to another (i.e. in chaining mode schemes), or between the entity that provides the data and the environments (i.e. in independent mode scheme). As a result, the data must be consciously known by the evidence subject [1], who can then evaluate whether the received data and that has been processed by other environments corresponds to the desired data. Obviously, an attacker can modify the visualization of fraudulent data in a certain environment to force its matching with the expected data. But the division principle guarantees that, if sufficient environments are used, a malicious action during evidence establishment will be detected with a probability substantially higher than if only one environment was used.

7.2.6.2 Restricted format of data to be signed

In the taxonomy proposed in Chapter 6, attack categories *D2-CAT2.1.1: Dynamic content inclusion* and *D2-CAT2.2.1: Dynamic content inclusion* cover attack methods that include dynamic content into the document or attributes to be signed, what permits the attacker to maintain the document's integrity while varying its semantic. On the other hand, attack category *D2-CAT3.1: External content* includes methods that permit an attacker to modify information referenced from the signed information, like the XSD or other documents.

As document formats are getting more and more complex, the capabilities of producing semantically different representations of the same document increase. Hidden code, active code or linked content are features that many document formats permit nowadays. An attack could very easily trick the evidence subject during the conscious

[1] An end user according to the thesis context

data verification by performing attacks *D2-CAT2.1.1: Dynamic content inclusion*, *D2-CAT2.2.1: Dynamic content inclusion* or *D2-CAT3.1: External content*, undermining the improvement pursued by the division principle.

We claim that no document with a complex or rich format can guarantee its semantic integrity. Therefore, in addition to the division of the signature environment, the format of the data to be signed should be restricted to static file formats, such as plain ASCII.

7.3 Chapter Summary

We claim that an untrustworthy environment cannot generate reliable signatures, and thus cannot enforce the non-repudiation of digital signature-based evidence. A single signature environment will have a higher or lower probability of suffering an attack, but the probability is never zero. As every new proposal is always welcomed with a new attack, a completely different approach must be taken.

Under the division paradigm, non-repudiation evidence consists of several signatures generated at different environments. In the same manner, the verification of a multi signature-based evidence must be performed at several environments, increasing the level of confidence in the verification result. The security mechanisms and additional issues for a practical implementation have been detailed. Our proposal does not require a change in current hardware architectures nor trust in third parties for the generation of evidence. It can be implemented entirely in software, complying with current pervasive computing necessities. In addition, the signature division paradigm could be combined with other security enhancing proposals (e.g. usage of a TPM architecture in each environment certified according to a certain Common Criteria Evaluation Assurance Level) to increase the reliability of evidence.

The division paradigm may be specifically applied to a wide variety of Internet protocols, like e-commerce or contract signing protocols, where several signatures must be performed by each participant.

Chapter 8

Extended Electronic Signature Policies

In this Chapter a new electronic signature policy is presented, and that extends the boundaries of the current electronic signature policy definition in a way that several signatures generated under a single transaction can be managed, and their relationships established. This new extended signature policy meets a current necessity and provides a solution for a general problem that cannot be covered by current policy definition. In addition, our extended policy resolves the particular problem derived from the division paradigm, permitting its implementation.

The extended electronic signature policy is defined in Section 8.1 using ASN.1 notation (see Appendix E for the XML-based definition). The procedures required for the generation and verification of signatures according to the extended signature policy are given in Section 8.2. This Section also includes the guidelines and proposals for the integration of the extended policy into current standards. Finally, the Chapter is concluded in Section 8.3.

8.1 Policy Definition

This Section proposes an extended signature policy (ext-SP) that allows the management of a set of signatures generated in a single transaction. We have taken the ETSI technical report on signature policy for extended business model [72] as the reference document that collects the high level requirements.

Though our proposal covers the most important aspects contained in [72], we consider that some of them cannot be transposed to an automatically processable document (e.g. ASN.1, XML). For instance, the umbrella approach outlined in Section 10.4 of [72] describes the signature policy and technical rules (Section 10.4.2) that are almost impossible to define in generic data structures. They cover so many different business and application domains that their specification should be done in free text form documents rather than in formal languages.

The ext-SP has been designed taking into account three different levels of abstraction:

Business Level. The first level defines the business and transactional contexts that apply to the signatures generated according to this policy.

Inter-relationships Level. The second level establishes the signatures that must be present in order to give legal effectiveness to the transaction as well as the relationships and dependences that are accepted among those signatures.

Atomic Level. In the third level, the requirements to be fulfilled by each signature on its own are defined. In practice, this level is implemented by current signature policies.

The definition of the ext-SP is contained in next Sections. For each data structure, and for clarity purposes, only the ASN.1 type definition and an example (if it contributes to understand the definition) are given. Appendix E contains both the ASN.1 module and the XML Schema Definition (XSD). Please refer to them for the analog XML elements definitions.

From here on, we refer to current signature policy definition [71, 203] as *signature policy*, and to the policy proposed herein, the *extended signature policy* or ext-SP.

8.1.1 Base structure

The next ASN.1 type defines the base of the ext-SP, and consists of the field *extSignPolicyInfo* and the field *extSignPolicyProtection.*

```
ExtSignaturePolicy ::= SEQUENCE {
  extSignPolicyInfo       ExtSignPolicyInfo,
  extSignPolicyProtection ExtSignPolicyProtection OPTIONAL
}
```

The whole information about the ext-SP is collected in the *extSignPolicyInfo* field, which ASN.1 type is the next one:

```
ExtSignPolicyInfo ::= SEQUENCE {
  extSignPolicyIdentifier ExtSignPolicyIdentifier,
  extSignValidationPolicy ExtSignValidationPolicy,
  extSignContext          [0] ExtSignContext OPTIONAL,
  extSignPolExtensions    [1] SignPolExtensions OPTIONAL
}
```

On the other hand, *extSignPolicyProtection* field includes the information about the cryptographic algorithm applied to protect the ext-SP. If this field is not included in the ext-SP, then an external protection mechanism should be used by the parties when transmitting the ext-SP through insecure means. More specifically, *ExtSignPolicyProtection* type is defined as follows:

```
ExtSignPolicyProtection ::= SEQUENCE {
  protectionAlg AlgorithmIdentifier,
  protection    BIT STRING
}
```

The protection algorithm, defined in *protectionAlg* field (*AlgorithmIdentifier* ASN.1 type, as defined in [57]) must be applied on the DER (Distinguished Encoding Rules) encoding [124] of the *extSignPolicyInfo* field. Different cryptographic algorithms could be used, like hash functions or digital signature algorithms. The ext-SP shall be protected by other means if the applied protection algorithm does not suffice in certain circumstances - a hash function does not prevent an attacker from modifying the ext-SP content if it is transmitted through insecure means, like a TCP/IP connection without SSL/TLS. If a digital signature algorithm is used (e.g. sha1withRSAEncryption), then the digital signature value will be encoded in the *protection* field. In this case, the digital certificate that wraps the public key corresponding to the signing private key must be provided by other means. The *subjectDN* field of the certificate should correspond to the *policyIssuerName* further defined.

The next Sections describe the fields indicated in the *ExtSignPolicyInfo* ASN.1 type. It should be noted that, following current standards philosophy, an optional extension field named *extSignPolExtensions* of *SignPolExtensions* ASN.1 type (as defined in [73, 203]) is included for future needs in most of types defined herein.

8.1.2 Policy identifier

The ext-SP must be uniquely identified by both signers and verifiers. The *extSignPolicyIdentifier* field of *ExtSignPolicyIdentifier* ASN.1 type is included for that purpose:

```
ExtSignPolicyIdentifier ::= SEQUENCE {
  extSignPolicyId        ExtSignPolicyId,
  dateOfIssue            GeneralizedTime,
  policyIssuerName       GeneralNames,
  extSigPolicyQualifiers [0] SEQUENCE SIZE (1..MAX) OF SigPolicyQualifierInfo
                         OPTIONAL,
  extSignPolExtensions   [1] SignPolExtensions OPTIONAL
}
```

The *extSignPolicyId* field is an Object Identifier (OID) that uniquely identifies this ext-SP among all policies issued by the issuer identified by *policyIssuerName* field. The *dateOfIssue* field indicates the date when this policy was issued. Finally, the *extSigPolicyQualifiers* field includes additional qualifying information, like the location where the ext-SP can be retrieved from. Its *SigPolicyQualifierInfo* type is defined in [74, 187].

An example of *ExtSignPolicyIdentifier* is given next:

```
extSignPolicyIdentifier {
  extSignPolicyId = { 1 0 }
  dateOfIssue = 19970717103000
  policyIssuerName[0] {
    uniformResourceIdentifier = jlopez.thesis.uc3m.es/Issuer
  }
}
```

8.1.3 Validation policy

The field *extSignValidationPolicy* of *ExtSignValidationPolicy* ASN.1 type is the core of the ext-SP, and describes the rules and conditions to be fulfilled by the set of signatures in order to give effectiveness to the transaction.

```
ExtSignValidationPolicy ::= SEQUENCE {
  signingPeriod         [0] SigningPeriod,
  treesOfSolutions      [1] TreesOfSolutions,
  extSignPolExtensions  [2] SignPolExtensions OPTIONAL
}
```

The *signingPeriod* field is of SigningPeriod ASN.1 type (as defined in [73, 203]), and identifies the period of time before and after which the ext-SP should not be used for creating signatures under this policy. It should be noted that a set of signatures created under a valid ext-SP can still be verified against the policy after its expiration date.

The *treesOfSolutions* field, further detailed, contains a set of graphs where each one represents a tree of signatures that defines the dependences and relationships among them. This field implements the **Inter-relationships Level** mentioned at the beginning of Section 8.1.

An example of *ExtSignValidationPolicy* is given next:

```
extSignValidationPolicy {
  signingPeriod {
    notBefore = 19970717103000
    notAfter = 20170717103000
  }
  treesOfSolutions[0][0] {
    ...
  }
}
```

Trees of Solutions

Taking into account the three types of signatures (parallel, sequential, embedded), it is obvious that a tree can be derived from the generated set of signatures. A tree is a connected graph with n vertices (nodes) and $n-1$ edges, and thus where there are no cycles. In particular, the tree used to represent the set of signatures has the next specific properties as well:

- The tree is a rooted tree in which the root node (level 0) represents the original signed document and the rest of nodes correspond to signatures.
- The edges have a natural orientation away from the root. The tree expands from the root towards the leaf nodes, which are nodes with no child.
- The graph is unweighted, that is, there are no edge weights.
- The tree is irregular: each node (signature) not being a leaf node can have a different positive degree, that is, it can have as many children as needed. Leaf nodes have positive degree 0.
- Every node has a negative degree 1 (number of parent nodes), except the root node which has negative degree 0.

Figure 8.1 depicts a generic set of signatures in tree form. Signatures in level 1 of the tree correspond to Primary Signatures (PS), which are either parallel or sequential signatures. The rest of signatures correspond to CounterSignatures (CS), which are embedded signatures that can be applied to either a PS or another CS. Besides, signatures which are children of the same parent behave as PS among them. The difference is that the signatures are applied to another signature instead of the document. Though the arrows follow a top-down direction (for search purposes), a signature in level n is applied to the parent signature in level $n-1$.

This first approach would seem valid to implement any multisignature based transaction. However, we have noticed that, in certain scenarios, more than one tree of signatures can make the same transaction be legally effective. It is the case of fair exchange [95, 197] and fair non-repudiation [142] protocols, where the transaction can be

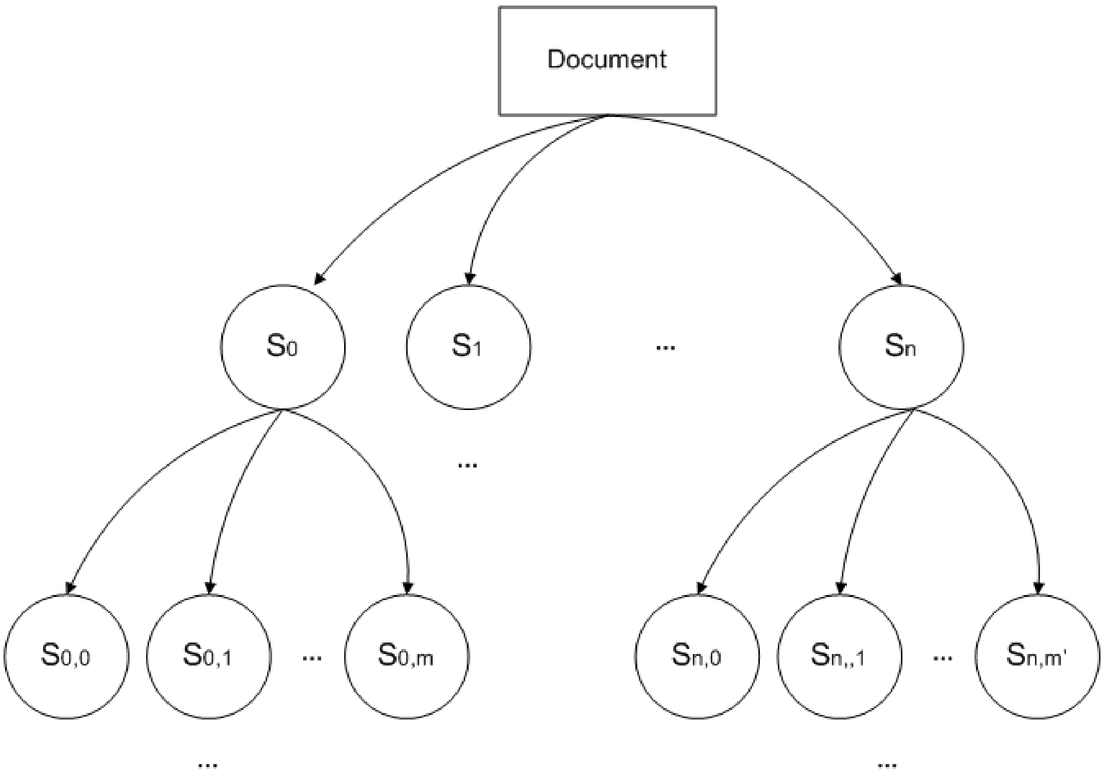

Figure 8.1: A generic tree of signatures.

finished (become effective) either by completing the main protocol or one of the specified subprotocols. Many other scenarios can be customized to follow this approach. For instance, an e-commerce protocol where two parties must sign a document can decide to either countersigning each others signature or let an authorized e-notary to do it.

In order to support this type of transactions, the Trees of Solutions (TSo) consists of a sequence of trees, each of which (named TSi) as represented in Figure 8.1. During the validation stage (see Section 8.2.2), the verifier must check if the multisignatures evaluated match one of the trees defined in TSo. The transaction is made effective providing that one tree is completely satisfied.

```
TreesOfSolutions ::= SEQUENCE OF treeOfSignatures TreeOfSignatures

TreeOfSignatures ::= SEQUENCE OF signature Signature
```

The *Signature* ASN.1 type defines the information of a particular node (signature) in a tree of signatures:

```
Signature ::= SEQUENCE {
  identifier             INTEGER (0..MAX),
  signer                 INTEGER (0..MAX),
  acceptableSignPolicies AcceptableSignPolicies,
  allowedCommitmentTypes SelectedCommitmentTypes,
  counterSignatures      [0] TreeOfSignatures OPTIONAL,
  timingAndSequence      [1] TimingAndSequence OPTIONAL,
  extSignPolExtensions   [2] SignPolExtensions OPTIONAL
}
```

Each node (signature) is uniquely identified by the *identifier* field. This information is used to specify timing and sequence dependences, as shown further. The signer that must perform this signature is uniquely represented in a figurative sense by the *signer* field. No signer specific information can be used (e.g. subject distinguished name or subject alternative name) as the ext-SP issuer does not know a priori which signer will actually perform the signature. A signer can appear in as many signatures of the tree as needed, but signatures to be generated by different signers must contain different *signer* values.

The *acceptableSignPolicies* field contains the signature policies (SP) OIDs [73, 203] that can be used by the signer when creating this signature. This field implements the **Atomic Level** mentioned at the beginning of Section 8.1.

```
AcceptableSignPolicies ::= SEQUENCE OF signPolicyId SignPolicyId

SignPolicyId ::= OBJECT IDENTIFIER
```

The *allowedCommitmentTypes* field restricts the commitment types that can be assumed by the signer when producing this specific signature. This field is of *Selected-CommitmentTypes* ASN.1 type (as defined in [73, 203]). These commitment types must be consistent with those included in the acceptable signature policies herein indicated.

Each signature can have a finite number of child nodes, which are represented by a *TreeOfSignatures* in the *counterSignatures* field. As a result, the tree is represented by following a recursive method, in which the leaf nodes of the tree will not have the *counterSignatures* field.

The time frame during which a signature must be generated and the sequential relationships with other signatures are described in *timingAndSequence* field.

```
TimingAndSequence ::= CHOICE {
  absoluteTimingAndSequence [0] SigningPeriod,
  relativeTimingAndSequence [1] SEQUENCE OF RelativeTimingAndSequence
}
```

The *TimingAndSequence* type supports the specification of sequential signatures. It allows any signature to have as many timing and sequence dependences on other signatures as needed. There are three possibilities:

- A signature has no actual dependence on any other signature (e.g. primary signatures).
- A signature has no dependence on other signatures but it must be performed within a period of time (e.g. a primary signature to be performed not before 17/07/1997 00:00:00 GMT and not after 17/07/2007 00:00:00 GMT). We define this dependence as an absolute dependence.
- A signature has certain dependences on other signatures, either sequential or embedded. These are considered as relative dependences.

The first case is achieved by omitting the *timingAndSequence* field of *Signature* type above. The second case is implemented by selecting *absoluteTimingAndSequence* field in *TimingAndSequence* type. To define one or more relative dependences, the *relativeTimingAndSequence* field must be selected, which ASN.1 type is the next:

```
RelativeTimingAndSequence ::= SEQUENCE {
  pathToRefSignature SEQUENCE OF INTEGER,
  maxDelta           DeltaTime OPTIONAL
}
```

The *pathToRefSignature* field indicates the path of node *identifier* field values from a signature located in level one of the tree to the signature with which there is a timing and sequence dependence. The *maxDelta* field indicates the maximum time delay allowed from the referenced signature's signing time during which this signature can be performed. That is, this signature must be performed in a period of time defined by $[t0, t0 + maxDelta]$ where $t0$ is the referenced signature's signing time. If this field is omitted, it means that this signature must be generated after the referenced signature but with no time limit.

In order to obtain accurate and reliable time references, signatures should be time stamped, following the requirements specified in the *TimestampTrustCondition* ASN.1 type [73, 203].

An example of *Signature* with a relative timing and sequence dependence is given next. In this example, the signature is the fifth node of the first *TreeOfSignatures* contained in the *TreesOfSolutions*:

```
treesOfSolutions[0][4] {
  identifier = 5
  signer = 1
  acceptableSignPolicies[0] = { 1 0 }
  allowedCommitmentTypes[0] {
    recognizedCommitmentType {
      identifier = { 1 2 840 113549 1 9 16 6 1 }
      fieldOfApplication {
        printableString = Proof of Origin }
    }
  }
  allowedCommitmentTypes[1] {
    recognizedCommitmentType {
      identifier = { 1 2 840 113549 1 9 16 6 4 }
      fieldOfApplication {
        printableString = Proof of Sender }
    }
  }
  timingAndSequence {
    relativeTimingAndSequence {
      pathToRefSignature[0] = 2
      pathToRefSignature[1] = 3
      pathToRefSignature[2] = 4
      maxDelta {
        deltaSeconds = 0
        deltaMinutes = 15
        deltaHours = 0
        deltaDays = 0
      }
    }
  }
}
```

8.1.4 Business and transactional domains

The context in which the extended signature policy applies is defined in the field *extSignContext* of *ExtSignPolicyInfo* ASN.1 type. This field is of *ExtSignContext* type, and implements the **Business Level** mentioned at the beginning of Section 8.1:

```
ExtSignContext ::= SEQUENCE {
  businessApplicationDomain [0] SigPolicyQualifierInfo OPTIONAL,
  transactionalContext      [1] SigPolicyQualifierInfo OPTIONAL,
  disputeResolution         [2] SigPolicyQualifierInfo OPTIONAL,
  audienceConditions        [3] SigPolicyQualifierInfo OPTIONAL,
  extSignPolExtensions      [4] SignPolExtensions OPTIONAL
}
```

The *businessApplicationDomain* field outlines the business domain in which the ext-SP is suitable for use, e.g. sale of goods/international trade transactions, e-Government transactions between citizens and e-Administration, e-health services, etc. It covers high-level and sector-oriented domains. On the contrary, the *transactionalContext* field provides additional information about the transactional context (e.g. draft of a contract, purchase by means of online service, exchange of design documents, etc.). This information should match with the *fieldOfApplication* field of each signature policy, described in *SignPolicyInfo* ASN.1 type [73, 203].

Disputes on a specific event or action taken by any party in a transaction may arise in a future. A dispute must be resolved by a third party with authority to do so, taking as information for the resolution the evidence generated in the transaction. Electronic signatures may act as non-repudiation evidence if adequate policies used by the parties enforce them. In that case, the third party must consider if the conditions established for the transactions have been fulfilled by the parties. The dispute resolution procedures are contained in the *disputeResolution* field. It allows the ext-SP issuer to specify a binding text to be considered by the parties when using this policy for generating and validating signatures, and by the third party for resolving a dispute.

Finally, *audienceConditions* states the conditions under which a signature may be relied upon, e.g. the signature only valid in a specified jurisdiction, where laws exist which recognize the legal validity of signatures created under conditions as specified in the policy. This field may include provisions relating to the intended effectiveness of signatures, where multiple signatures are required, e.g. the signature must be countersigned to be relied upon.

Each field (except the extension field) is of *SigPolicyQualifierInfo* ASN.1 type, defined in [74, 187]. Therefore, the information for each field could be available at a Web URI or URL reference (specified by *SPuri* type of qualifier), or explicitly contained in the qualifier through the *SPUserNotice* qualifier, which may contain a reference to the organization notice and an explicit text. Please refer to Section 5.8.1 of [74, 187] for further information.

An example of *ExtSignContext* is given next.

```
extSignContext {
  businessApplicationDomain {
    sigPolicyQualifierId = { 1 2 840 113549 1 9 16 5 1 }
    sigQualifier {
      explicitText {
        visibleString = Sale of goods/international trade transactions }
    }
  }
  transactionalContext {
    sigPolicyQualifierId = { 1 2 840 113549 1 9 16 5 1 }
    sigQualifier {
      explicitText {
        visibleString = Purchase Order/Acceptance in relation to
        a book purchase order made through Alice Bookshop Internet
        Web page between Alice Bookshop and a client of Alice Bookshop }
    }
  }
  disputeResolution {
    sigPolicyQualifierId = { 1 2 840 113549 1 9 16 5 1 }
    sigQualifier {
      explicitText {
        visibleString = Any disputes arising under this policy
        shall be referred to a suitably qualified expert, whose
        decision shall be final and binding upon the parties,
        provided that this signature policy imposes the constraints
        under which any signature created under it shall be valid.
        The dispute resolution procedure shall be carried out in a
        European court with appropriate responsibilities }
    }
  }
  audienceConditions {
    sigPolicyQualifierId = { 1 2 840 113549 1 9 16 5 1 }
    sigQualifier {
      explicitText {
        visibleString = The digital signature-based evidence is
        only valid in a specified jurisdiction, where laws exist
        which recognize the legal validity of signatures created
        under conditions as specified in the policy }
    }
  }
}
```

8.1.5 Signing roles

A signing role is a role allocated to or adopted by a signer, and which defines the relationship between its signature and the rest of signatures [72]. A signing role is mainly a Primary Signature (PS) or a CounterSignature (CS).

In our proposal, the signing role is implicitly assigned to a signer by means of the position that his corresponding signature has in the concrete tree of signatures (TSi). Therefore, a signature mapped to a node in level one of TSi implies that the signer is assuming a PS signing role. Otherwise, the signing role is a CS. This behavior completely satisfies the requirements respecting signing roles given in Section 9.4.1 of [72].

8.2 Using the Policy

This Section deals with the steps a signer – Section 8.2.1 – and a verifier – Section 8.2.2 – must follow in order to adhere to this policy. We suppose that a signature application that supports the ext-SP is available to both signers and verifiers, the ext-SP can been retrieved by the application, and its integrity, authenticity and validity period verified. Also, the acceptable signature policies indicated in the *acceptableSignPolicies* field of each signature can be retrieved and verified as well.

Finally, the way the extended signature policy can be integrated into current standard signature formats is given in Section 8.2.3.

8.2.1 The generation process

A proposal for the procedure a signer must follow in order to generate a set of signatures (SSi) according to the ext-SP is detailed in the next steps. We suppose that the signer is an end user that owns a device with cryptographic capabilities and a visualization screen for data representation (e.g. Personal Computer, mobile device, etc.). Some actions further explained could be skipped in case of automated signing processes, like those carried out by proxy servers and backend systems, or when the business context does not require human interaction (e.g. a B2B transaction).

Step 0 - Selection of the data to be signed and policy to use

This step has to be carried out only once by the first signer (or the process in charge of that), as subsequent signatures will be either Primary Signatures performed over the same data or CounterSignatures generated over another signature. From that moment onwards, each signer will be shown the signed data without the possibility of changing or adding information during the signing process.

Step 1 - Policy information visualization and pre-processing

As a first step, the signer should be shown the ext-SP information, specially the Tree of Solutions (TSo) data. Because the TSo may contain several Trees of Signatures (TSis), the application should show every TSi defined therein. The application should also permit the signer to visualize all the information of each node of the TSis, including the acceptable signature policies, the allowed commitment types and the timing and sequence dependences. Furthermore, the information of the business and transactional domains contained in the *extSignContext* field of the ext-SP should be displayed to the signer.

If a partially generated SSi is already available, the application must show the relevant information of each signature (e.g. signatory, signing time, validation information, etc.) in a manner that a mapping between the structure of the trees represented by the TSis and the tree represented by the SSi can be intuitively traced by the signer.

If no SSi is present (it is the first signer), then every node in the first level of the TSis must be marked as signable. That is, the signer is, a priori, able to perform any of the Primary Signatures defined in the TSis.

On the other hand, if a SSi is present, then the application should filter those TSis that are not fulfilled by the SSi. For the rest of TSi (if none, then the pre-validation fails, and the signer must not be allowed to perform any signature), and as a result of the partial validation that must have been carried out by the application, the signatures present in that moment and the signatures that are left to complete the transaction should be differentiated. In a nutshell, the TSis that are active are those that, until then, are being fulfilled by the partial SSi. Potentially signable nodes of these TSis are those not signed yet and that belong either to the first level of the TSis (Primary Signatures) or to any other level (CounterSignatures) but are children of an existent signature. It is worth to remember that the signer will be bound to the data selected by the first signer (see step 0).

It should be mentioned that potentially signable nodes that do not comply with the timing and sequence dependences must be discarded. For instance, a node must not be marked as signable if, though conditions above are met, a signature on which the one to be generated depends on has not been performed yet. Or, if, for example, the time interval during which the signature must be generated (defined by *NotBefore* field of an absolute dependence) is later than the current time. Moreover, if during the ext-SP pre-processing the application detects that the timing requirements of a TSi node are impossible to be fulfilled (e.g. the node indicates that the signature must be generated before 17/07/2007 00:00:00 GMT, but the current date is 17/07/2007 00:15:00 GMT), then the pre-validation must fail, and the signer must not be allowed to perform any signature.

Step 2 - Certificate selection

The signer should then be asked to select the digital certificate to use. The certificate must be selected among available certificates with associated signing private key, and both software certificates and certificates stored in a hardware cryptographic device should be supported by the application. Nonetheless, it is the signature policy selected by the signer (in step 4) the one which will restrict the usage of one type of certificate or another. If the selected policy does not accept the certificate, then the whole process from step 2 to 4 has to be carried out again. Notwithstanding, the extended policy may also restrict at first instance the certificate to use according to the division and binding principles explained in Chapter 7. For example, the extended policy may forbid the signer to select a certificate already used in a previous signature of the partial SSi, if present.

Step 3 - Candidate nodes

Once the certificate has been chosen, the application must select the nodes that potentially represent the signature to be generated by the signer. These candidate nodes will be among the signatures that have been previously marked as signable, and can be from more than one TSi.

The application must extract the *subjectDN* information from the certificate. This information represents the digital identity of the signer. Afterwards, the application must detect if the signer has already performed any signature in the SSi tree. In that case, the association *subjectDN* and *signer* identifier can be done, and the candidate nodes be easily highlighted (see Section 8.2.2 for further information about the association and matching processes). Otherwise, there may be several possible *signers* to which the *subjectDN* can be associated. The candidate nodes are highlighted anyway.

Step 4 - Node selection

In this step the signer must select the candidate node over which applying the signature. The data to be signed can be the document (root node) or another signature (see step 0). The signer must also select the type of commitment to make among those available in the selected node. The application will then allow the signer to select a signature policy among the policies in the node (*acceptableSignPolicies* field) that support the selected commitment type.

Step 5 - Signature computation

Finally, the signer performs the signature. The signature generation process is carried out according to the signer rules of the selected signature policy, as stated in the corresponding *signerRules* field (see [73, 203]).

It should be noted that once the signature is created and added to the SSi, the number of TSis that are still satisfied by the current SSi may be reduced.

The signer can abort the generation process at any time before step 5. If, at step 1 or 3, the application obtains a void set of signable or candidate nodes, respectively, then the generation process must be automatically aborted. This situation implies that either the set of signatures does not comply with the ext-SP requirements or the signer cannot generate a compliant signature at that moment based on the selected certificate.

It is important to emphasize that every signature that is already present must be verified against the ext-SP before allowing the signer to compute the signature. In the description above, the procedure assumes that it is the application the one in charge of pre-validating the partial SSi against the ext-SP. However, the verification procedure could be delegated to a trusted party, if possible. In any case, the party in charge of verifying the set of signatures must verify the already existent signatures and collect the required validation data as specified in the verifier rules section of each signature policy (see [73, 203]).

Finally, the generation process could be automated for simplification purposes, if necessary, on a case-by-case basis.

8.2.2 The validation process

During the validation stage, the verifier will check if the set of signatures fulfills the requirements established in the referenced ext-SP, and if each signature is compliant with its corresponding referenced signature policy.

The next Section 8.2.2.1 provides an overview of the designed validation strategy. Section 8.2.2.2 describes the pruning methods that have been incorporated. A refinement stage needed to complete the validation process is explained in Section 8.2.2.3. Appendix C contains the pseudo-code of the algorithm along with the details of the validation strategy.

8.2.2.1 Approach

Each Tree of Signatures (TSi) belonging to the Trees of Solutions (TSo) and the generated set of signatures (SSi) represent a tree graph with the properties explained above in Section 8.1.3. Our validation process combines a Depth-First-Search (DFS) algorithm to explore the SSi and a modified Breadth-First-Search (BFS) algorithm to locate each signature of SSi in a TSi. Other strategies different than the proposed one may achieve the same aim, which is to evaluate if SSi can be mapped to at least one TSi, what would imply that SSi is compliant with the ext-SP. Furthermore, four different pruning methods that improve the process performance and effectiveness have been integrated.

Graph search algorithms like BFS or DFS follow an established strategy to explore a graph or search a node in a graph. In particular, BFS starts at the root node and explores all the neighboring nodes. Then, for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until the graph has been completely explored or the node is found. DFS starts at the root node and explores along each branch until the leaf nodes before backtracking. It continues until the whole graph has been explored or the goal node found.

BFS and DFS are combined and modified in order to cope with next particularities:

- We need to search as many nodes in TSi as signatures in SSi, not just one signature (node).
- Each signature in SSi can be found only in a specific level of TSi. Therefore, we can apply an important filter in the search strategy. As an example, signatures located in first level of SSi will be searched only in TSi nodes in level one. Their children signatures (countersignatures) will be searched only in TSi nodes in level two, and so forth.
- There is a natural reduction in the number of searching possibilities for a particular signature. The reason is twofold: a signature can be found in a specific level of TSi but only in those nodes that are children of nodes matching the parent signature. Besides, a reduction can be applied to the potential search paths by means of several types of pruning (see Section 8.2.2.2). The nodes of TSi used by the algorithm in order to find a signature are called candidate nodes.
- A signature can be matched (found) with more than one candidate node in the specific level of TSi. Therefore, our algorithm does not stop when a signature is matched, but when all candidate nodes have been explored. The reason is that different signers can make the same commitment type and use the same signature policy when creating their signatures. As a consequence, there is no way of differentiating which node of TSi better matches with each signature, except when a pruning is performed (see Section 8.2.2.2).

As a result, SSi is explored by using a DFS strategy that will visit every signature. Each time a signature is visited, the algorithm tries to match it with the corresponding candidate nodes of TSi. The candidate nodes are explored by following a BFS strategy, but with the particularities explained above.

Each signature is represented by a set of information that allows the algorithm to perform the search. These search parameters are the next:

- The **subject distinguished name**, which can be retrieved from the *SubjectDN* field of the signer's certificate.

- The **signature policy** (OID value) used by the signer when creating the signature. This value can be obtained from the policy identifier, included in the signature as a signed attribute. Please refer to Explicit Policy-based Electronic Signatures (EPES) formats in [74, 187] for further information.
- The **commitment type** (OID value) made by the signer respecting the signed information. This information is also included in the signature as a signed attribute. Please refer to EPES formats in [74, 187] for further information.

On the other hand, the TSi node information used by the algorithm for the search is the following:

- The **signer** field of a node, which is a representation of the signer of the node.
- The **acceptable signature policies**, which OIDs are contained in the *acceptableSignPolicies* field of the node.
- The **allowed commitment types**, which OIDs are contained in the *allowedCommitmentTypes* field of the node.

Therefore, the algorithm will try to match a signature with the candidate nodes by using previous search parameters and node information.

Next three end conditions for the algorithm apply:

- If a TSi is completely satisfied after having processed SSi, then SSi represents a solution, and the transaction is made effective and considered complete.
- If after having completely processed SSi, there is at least one TSi that is partially satisfied (no deadlock is found), then SSi represents a partial solution that may become complete if the rest of required signatures are generated. We define a *deadlock* as the situation where a signature cannot be matched with any candidate node, and as a result the search algorithm cannot continue processing the SSi.
- Otherwise, SSi is not compliant with the ext-SP conditions, and the validation fails.

Definition 1. Providing that the number of signatures in SSi and nodes in TSi are the same, and the structure of both graphs is the same (homomorphic graphs), a TSi is completely satisfied if, once the validation algorithm is finished, each signature of SSi is matched with at least one node in TSi and every node of TSi is matched with at least one signature of SSi.

Definition 2. SSi is compliant with ext-SP if at least one TSi is completely satisfied. The set of signatures can be complete - every signature needed to complete the transaction has been generated (TSi completely satisfied) - or incomplete - there are still some required signatures left (TSi partially satisfied).

8.2.2.2 Pruning methods

The aim of the pruning methods is to dynamically reduce during the search the number of possible nodes that can be matched with each signature. As a result of this on-the-fly refinement, the number of paths to explore in further steps can be substantially reduced, improving the computational cost and memory consumption. Besides, pruning methods can detect a deadlock situation that may not be detected otherwise, or detected in a later step.

Distribution-based pruning

As commented above, TSis and SSis are trees where each node can have a different positive degree (except leaf nodes that have positive degree 0) but all of them have the same negative degree 1 (except the root node that has negative degree 0). Furthermore, the information of a node – and thus a child node – (signer's identifier, the allowed commitment types and the acceptable signature policies) can be used to differentiate it from other nodes. In the same way, the information of a signature (subject distinguished name, commitment type made and signature policy used) can also be used to differentiate it from other signatures.

The distribution-based pruning makes use of these two facts to prune the candidate nodes that can potentially be matched with a signature being analyzed during the search.

Definition 3. We define the distribution of a node as the number of occurrences for each dimension (1...n), being a dimension the number of child nodes with the same identifier (signer identifier for TSi nodes and *subjectDN* for SSi nodes). That is, a dimension n means that there are n child nodes with the same identifier. We represent a distribution as a list of key pairs $\{1 = a, 2 = b, 3 = c, ...\}$ where the key corresponds to the dimension and the value is the number of occurrences of that dimension.

Consider the simple tree shown in Figure 8.2. In this example, the root node has four Primary Signatures, one of them with the *subjectDN* S_0, another two with *subjectDN* S_1 and the last one with *subjectDN* S_2. Therefore, the number of occurrences for dimension one is 2, as there are two *subjectDN* found in just one signature each (S_0

and S_2). On the other hand, the number of occurrences for dimension two is 1, as there is only one *subjectDN* that appears in two signatures (S_1). Consequently, the distribution for the root node is $\{1 = 2, 2 = 1\}$. As can be noted, the distribution of a node in a tree gives an exact view of its underlying children.

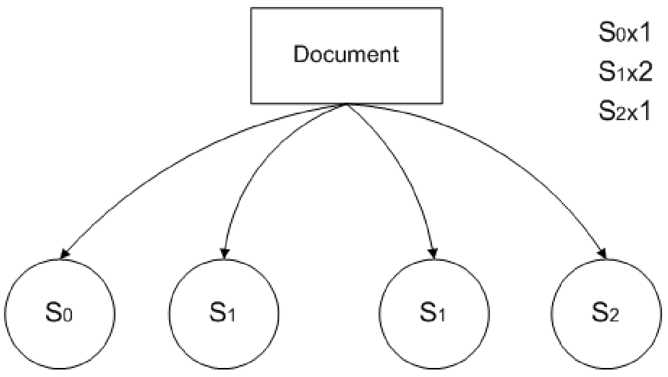

Figure 8.2: Distribution of a tree.

Definition 4. Two nodes are considered structurally equal if and only if their distributions are the same.

As a result, a signature cannot be matched with a node if their distributions differ. In order to filter the candidate nodes of TSi that can be matched with a certain signature of SSi, the algorithm calculates the distribution for each candidate node and for the signature being analyzed. Only those nodes which distribution matches the signature distribution are further processed. Obviously, if no candidate node has the same distribution as the signature, a deadlock occurs. In particular, if the distribution of the TSi root node differs from the SSi root node, then the SSi does not satisfy the TSi.

Dimension-based pruning

This pruning is related to the Distribution-based pruning. As explained above, the dimension corresponds to the number of nodes with the same identifier. In the example above, node S_1 has dimension two because its *subjectDN* appears twice.

Next Figure 8.3 shows the structure of child nodes for both TSi node n_8 and SSi signature S_7. In both cases, the distribution is the same: $\{1 = 3, 2 = 1\}$.

Suppose then that signature S_7 has been matched with node n_8, and thus their child nodes are analyzed during the exploration of the trees. When a distribution is calculated, the algorithm also works out the dimensions for the nodes. In the example of Figure 8.3, next dimensions are obtained for the second level (shown) of the trees:

- TSi: $\{1 = \{n_0, n_1, n_2\}, 2 = \{n_3\}\}$.

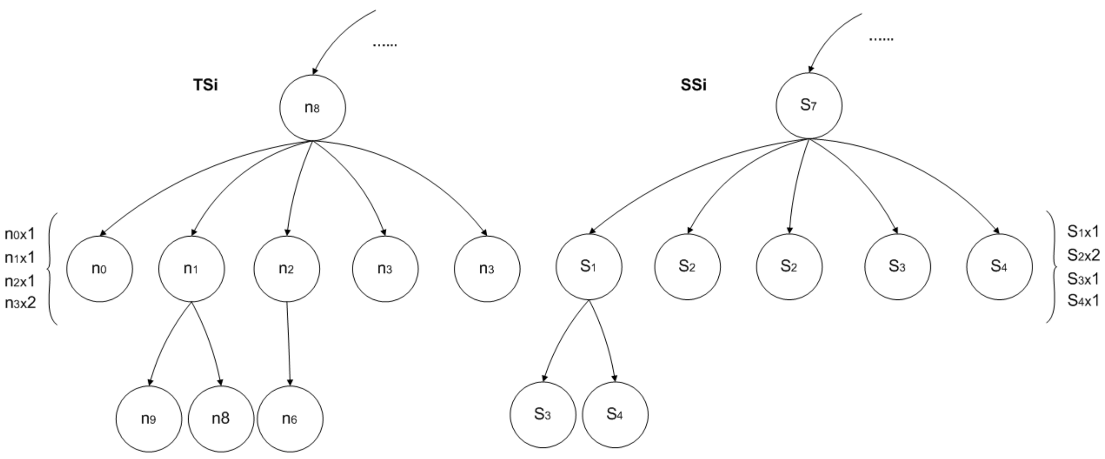

Figure 8.3: Example of a distribution used for a dimension-based pruning.

- SSi: $\{1 = \{S_1, S_3, S_4\}, 2 = \{S_2\}\}$.

Where the key is the dimension and the value is the collection of node signer's identifiers (if TSi) or *subjectDN* (if SSi) with that dimension.

The Dimension-based pruning lies in that the particular dimension of a SSi signature can be used to filter the candidate nodes by selecting only those which dimension coincides. For instance, the resultant candidate nodes for signatures S_1, S_3 and S_4 (dimension 1) are $\{n_0, n_1, n_2\}$, while for signature S_2 (dimension 2) is only $\{n_3\}$.

This pruning can be applied either before or after the Distribution-based pruning. As such, both complement each other. In our case, we have designed the algorithm to apply the Dimension-based pruning before the Distribution one. As an example, initial candidate nodes for signature S_1 are $\{n_0, n_1, n_2, n_3, n_3\}$. After applying the Dimension-based pruning, the resultant candidate nodes are reduced to $\{n_0, n_1, n_2\}$. Finally, and after applying the Distribution-based pruning, the final candidate node is $\{n_1\}$.

Signer-based pruning

The first time a *subjectDN* is used as a search parameter, its corresponding signature will be matched with one or more nodes of TSi in a certain level. As a result, the *subjectDN* will be associated to one or more *signer* identifiers. There are two possibilities providing that a deadlock is not found:

Non-definite signer assignment. If several matches have been made, the *subjectDN* cannot be assigned to a definite signer identifier. However, the algorithm can still use this list of assigned signer identifiers to filter candidate nodes in future steps.

Definite signer assignment. If there is only one match, then a definite signer identifier assignment is done. From that moment onwards, the *subjectDN* is linked to a unique signer identifier. This association is used by the algorithm to discard candidate nodes in future steps.

As an example of a non-definite assignment, suppose that a signature with *subjectDN* equals to *CN = Researcher, OU = Computer Science Department, O = University Carlos III of Madrid* is matched with two nodes in level 2 of TSi, the first node with signer's $id = 1$ and the second one with signer's $id = 2$. The pruning here lies in that future searches of signatures with the same *subjectDN* can only be matched with nodes with signer identifiers equal to 1 or 2.

On the other hand, suppose that a signature with a different *subjectDN* is matched, after having processed the candidate nodes, with just one node with signer's $id = 3$. In this case a definite signer assignment is done. Therefore, future processing of signatures with that *subjectDN* can only be matched with nodes with $id = 3$. Moreover, future processing of signatures with different *subjectDN* cannot be matched with nodes with $id = 3$.

Once a signature has been firstly matched with certain nodes at a specific level of TSi, the algorithm knows that the signer identifier to which that *subjectDN* will be finally assigned is one of those identifiers, and no other. Otherwise, the matching would imply a contradiction.

A signer-based pruning can also be applied while backtracking. For instance, consider the example shown in Figure 8.4:

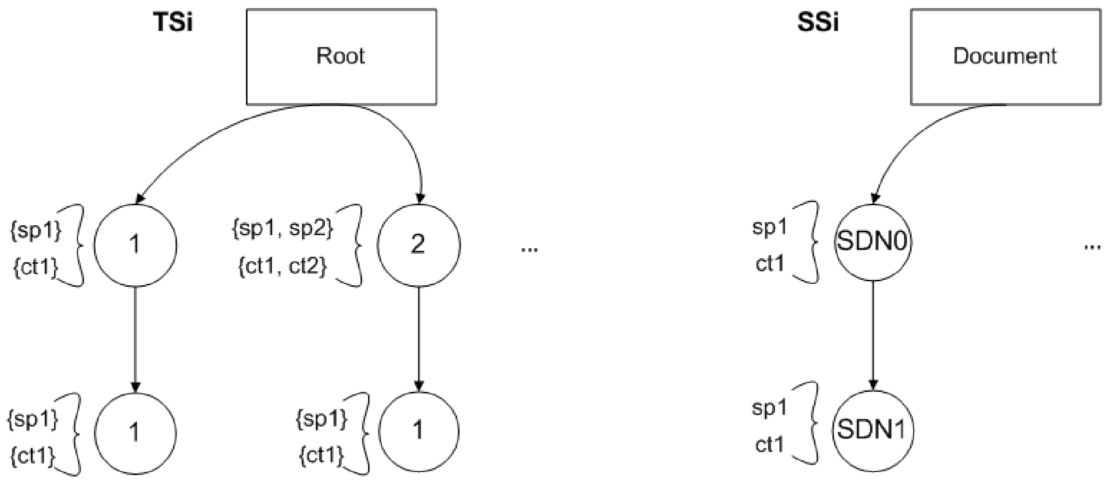

Figure 8.4: Identifier-based pruning example.

In Figure 8.4, the signature with *subjectDN SDN0* is matched with nodes with signer identifiers $id = 1$ and $id = 2$. While processing its countersignature, the signer identifier 1 is definitely assigned to a different subject distinguished name, in this case the *subjectDN SDN1*. As a result, while backtracking, the matching between the node with signer identifier 1 and the signature *SDN0* is undone. Moreover, derived from this unmatching, and as only one matched node remains, a second definite signer assignment is done between *SDN0* and signer identifier 2.

It should be noted that a deadlock situation would have occurred if signer identifier 2 had been previously and definitely assigned to a second countersignature with different *subjectDN*. Due to the signer-based pruning, the signature in the example would have ended without matched nodes, and the validation would have failed.

In order to be able to manage this pruning method, the algorithm must maintain an updated list of assigned signer identifiers (both definite and non-definite) that has to be looked up in each step.

Path-based pruning

As previously mentioned, the algorithm processes each signature focusing on its corresponding TSi search level. Once a signature has been matched with certain nodes, the countersignatures are processed in the next deeper level, but using as candidate nodes those that are children of nodes matching the parent signature, as explained in Section 8.2.2.1 above. This pruning method consists of the following: Once the leaf signatures are reached, the algorithm backtracks, providing a list of the parent nodes that resulted in a matching. Parent nodes not included in the list are discarded as matched nodes for the parent signature. As a result, the space of possibilities, that is, actual nodes which requirements are fulfilled by that signature, is reduced and refined.

For example, let T be a TSi of depth n, and S be a SSi of depth n as well. Let s_{n-1} be a signature of S in level $n-1$, and s_n be its countersignature in level n. If s_{n-1} is matched with nodes $node1_{n-1}$, $node2_{n-1}$ and $node3_{n-1}$ of T, then the algorithm uses nodes children of $node1_{n-1}$, $node2_{n-1}$ and $node3_{n-1}$ as candidate nodes for countersignature s_n. However, suppose that only a candidate node child of $node2_{n-1}$ produces a matching with countersignature s_n. As a result, when the algorithm backtracks, nodes $node1_{n-1}$ and $node3_{n-1}$ are discarded as matched nodes for s_{n-1}. In this case we say that a path-based pruning has been done.

A path-based pruning can result in a definite signer identifier assignment if, after having applied the path-based pruning, only one matched node remains.

Figure 8.5 illustrates an example where signer-based and path-based prunings are applied. Signer identifiers, acceptable signature policies and allowed commitment types of the TSi nodes are shown. The subject distinguished name, signature policy used and commitment type made of each SSi signature are shown as well.

Though it would seem at first glance that the SSi is compliant with the TSi, a deadlock occurs during the validation. In the first step, signature *SDN0* is matched with nodes *id1* and *id2* in first level, as the signature policy used and the commitment type made are among those permitted by both TSi nodes. When processing the first countersignature (*SDN1*), the algorithm evaluates the four child nodes in level 2 of TSi as candidate nodes. This countersignature is only matched with the first candidate node (*id1*). Then, a definite signer identifier assignment is done. As it is a leaf signature, the algorithm backtracks. At this point, the algorithm detects that only the path of the node *id1* in first level resulted in a matching. As a consequence, node *id2* in first level is discarded as a matched node for signature *SDN0*. Besides, the algorithm detects that the signer identifier *id1* has been definitely assigned to a different subjectDN (*SDN1*). Therefore, the matched node *id1* is discarded as well, and a deadlock occurs.

As can be seen, both pruning methods feed each other, improving the overall performance and accuracy of the algorithm.

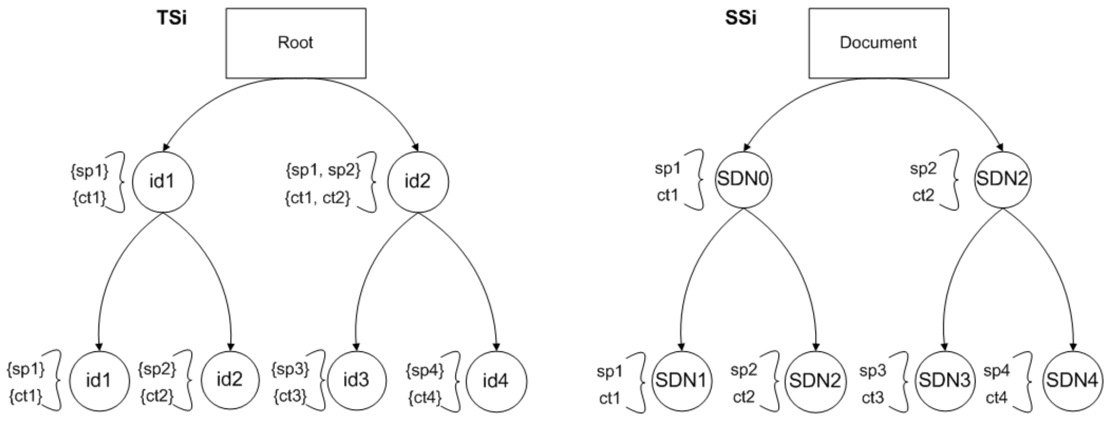

Figure 8.5: Deadlock example.

8.2.2.3 Refinement stage

Due to the followed DFS strategy, SSi is processed in a top-down (bottom-up when backtracking) and left-right manner. As a result, prunings and signer identifier assignments produced during the tree evaluation have no effect on the already processed signatures. For that reason, and providing that a deadlock has not occurred during the search, a refinement stage has to be applied before concluding the validation algorithm.

At this stage, the information generated during the search is analyzed. The information mainly covers the matches between signatures and nodes, definite and non-definite signer assignments, and timing and sequence dependences, which have not been evaluated so far.

The aim of this stage is threefold, consisting of three phases:

- Detect possible deadlocks not detected yet. A signer identifier can be definitely assigned to a *subjectDN*, but previously processed signatures with different *subjectDN* still maintain the same assignment. On the other hand, signatures may be matched with nodes which parent nodes do not derive in a matching for their right-hand side siblings (path-based pruning), but their matchings are not updated during the search. For these reasons, during this stage both pruning methods are iteratively applied until no change is produced, that is, a stable version of the solution is obtained or a deadlock is found. This process is called the **refinement phase one**.

- Analyze if every node of TSi is matched with at least one signature of SSi (SSi satisfies this TSi). This analysis is carried out in the **refinement phase two**.
- Finally, in the **refinement phase three**, the timing and sequence dependences among the signatures are evaluated. It is not possible to analyze these constraints until each signature has been matched with nodes. Therefore, this analysis must be done at this stage. Ideally, at this point each signature is matched with just one node, and the evaluation of the dependences is straightforward. However, sometimes there may be multiple matches for a signature. In these cases, each combination must be considered for the timing and sequence analysis until either a solution is found or every combination has been tested without producing a solution. In this case, the validation fails. It is important to remark that an ext-SP wrongly defined can provoke a deadlock (as traditionally defined in process scheduling) respecting the timing and sequence dependences. For instance, if a signature $s1$ must be generated after the signature $s2$, but $s2$ must be generated after $s1$ (impossible condition), then a deadlock occurs. The ext-SP issuer is responsible for defining a correct ext-SP.

As the very last step, and once a solution is found, each signature must be evaluated according to the requirements of the referenced signature policy. Only when this final validation step is successfully completed, the set of signatures can be said to comply with the extended signature policy. It should be mentioned that this verification could be performed before starting the validation algorithm as well.

8.2.3 Integration in AdES formats

Advanced Electronic Signature Formats (AdES) like CAdES [74, 187] and XAdES [59, 75] have adopted the inclusion of the signature policy reference as a signed attribute in their EPES version. By signing over the signature policy identifier, the signer explicitly indicates that he has applied the signature policy in creating the signature. The verifier is also able to retrieve the referenced signature policy content and thus validate the signature accordingly. In order to unambiguously identify the referenced signature policy that is to be used to verify the signature, the signed attribute includes an identifier unique in the domain of the signature policy issuer and a hash of the signature policy document.

In order to support the usage of extended signature policies in the same way, a new signed attribute has to be defined. For CAdES signatures, we propose the following *id-aa-ets-extSigPolicyId* object identifier (OID) to identify the new **extended signature policy identifier attribute**[1]:

```
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16)
  id-aa(2) 49 }
```

The attribute value has *SignaturePolicyIdentifier* ASN.1 type, as defined in [74, 187]:

```
ExtSignaturePolicyIdentifier ::= SignaturePolicyIdentifier

SignaturePolicyIdentifier ::= CHOICE {
  SignaturePolicyId      SignaturePolicyId,
  SignaturePolicyImplied SignaturePolicyImplied
}
```

On the other hand, and for XAdES signatures, a next signed property, child of *xades:SignedSignatureProperties* element and of type *SignaturePolicyIdentifierType* XML type (as defined in [75]), is proposed:

```
<xsd:element name="ExtSignaturePolicyIdentifier"
             type="xades:SignaturePolicyIdentifierType"/>

<xsd:complexType name="SignaturePolicyIdentifierType">
  <xsd:choice>
    <xsd:element name="SignaturePolicyId" type="SignaturePolicyIdType"/>
    <xsd:element name="SignaturePolicyImplied"/>
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="SignaturePolicyIdType">
  <xsd:sequence>
    <xsd:element name="SigPolicyId" type="ObjectIdentifierType"/>
    <xsd:element ref="ds:Transforms" minOccurs="0"/>
    <xsd:element name="SigPolicyHash" type="DigestAlgAndValueType"/>
    <xsd:element name="SigPolicyQualifiers"
                 type="SigPolicyQualifiersListType" minOccurs="0"/>
```

[1] At the time of writing the thesis, OID 1.2.840.113549.1.9.16.2.49 was not assigned

```
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SigPolicyQualifiersListType">
  <xsd:sequence>
    <xsd:element name="SigPolicyQualifier" type="AnyType"
                 maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

The signature would then include as signed attributes the signature policy and extended signature policy references along with the commitment type made by the signer.

8.2.4 Certificate extension

From the division principle perspective, the extended signature policy basically defines the signatures that are part of certain non-repudiation evidence. By incorporating the extended signature policy identifier signed attribute (see Section 8.2.3), it is being indicated that the corresponding signature is part of certain evidence, and that it cannot be enforced nor the signer can be held liable for any commitment made with such signature until all signatures are present. On the other hand, Section 7.2.5 of Chapter 7 presented two proposals to bind each environment with the signature creation data (SCD) that must be used within its boundaries. The joint application of the signed attribute and the environment-SCD binding assures the enforcement of the division principle. However, a problematic scenario may arise.

Let's assume that an attacker has compromised one of the environments needed to produce non-repudiation evidence. The environment-SCD binding assures that the attacker cannot gain access to every SCD required for the evidence composition unless all environments are compromised. But the attacker is still capable of generating digital signatures on behalf of the user by using the SCD bound to that environment. Such signatures could be used in certain non-repudiation service or have legal effectiveness, provided that the extended signature policy identifier signed attribute is not included, what is clearly reasonable once the attacker controls the environment.

To avoid an attacker to obtain a benefit from fraudulent signatures, the usage of every public key should be restricted to just the extended signature policy scope. Chapter 2 introduced the Public Key Infrastructure (PKI) and the digital certificate. An important field of a certificate, called *keyUsage*, permits to delimit the purposes or usages of the key embedded in the certificate. However, key usages defined in PKI are related to the cryptographic operations that can be carried out with such keys, like data origin authentication or key agreement.

In order to incorporate customized key usages, the standard defines the **extended key usage extension** field (*extKeyUsage*), by which one or more purposes for which the certified public key may be used can be indicated, in addition to or in place of the basic purposes indicated in the key usage extension. Using this field, we define the next extended key usage by which the purpose of the key is restricted to verify digital signatures generated under the scope of an extended signature policy[1]:

```
id-kp OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6)
internet(1) security(5) mechanisms(5) pkix(7) keyPurpose(3) }

id-kp-nonRepudiationExtSP OBJECT IDENTIFIER ::= { id-kp 18 }
-- Verify digital signatures under the scope of an extended electronic
-- signature policy
```

As defined by the standard [57], if this extension is present, then the certificate must only be used for one of the purposes indicated. Also, if multiple purposes are indicated, then the application need not recognize all purposes indicated as long as the intended purpose is present. However, and as stated by the standard, if a certificate contains both a key usage extension and an extended key usage extension, then both extensions must be processed independently and the certificate must only be used for a purpose consistent with both extensions. Consequently, we recommend that certificates issued to fulfill this thesis proposal must incorporate *nonRepudiation* key usage (renamed to *contentCommitment* in last version of the standard [57]) and *nonRepudiationExtSP* extended key usage.

Another alternative we propose herein consists of including a new certificate extension for X.509 v3, named **extended signature policy extension**. According to [57], certificate extensions provide methods for associating additional attributes with users or public keys. Our new extension would permit any PKI to limit the scope of the certificates issued, allowing only their usage in services orchestrated by extended signatures policies. Therefore, if the electronic signature being verified with the public key embedded in such certificate does not include a reference to an extended signature policy, then the signature must not be deemed valid.

This other proposal complies with current standard as it is indicated that the X.509 v3 certificate format allows communities to define private extensions to carry information unique to those communities. Each extension includes an object identifier (OID)

[1]At the time of writing the thesis, OID 1.3.6.1.5.5.7.3.18 was not assigned

and an ASN.1 structure. We propose the following OID for the new extended signature policy extension[1]:

```
id-ce   OBJECT IDENTIFIER ::=  { joint-iso-ccitt(2) ds(5) 29 }
id-ce-extSignaturePolicy OBJECT IDENTIFIER ::=  { id-ce 65 }
```

The attribute value has *SignaturePolicyIdentifier* ASN.1 type, as defined in [74, 187] (see also Section 8.2.3):

```
ExtendedSignaturePolicy ::= SignaturePolicyIdentifier
```

Both solutions given here are based on refining the semantic and usage of the certificate by means of certificate extensions. Any of them, the extended key usage extension or the extended signature policy extension, may be used indistinctly or in conjunction.

Extensions (e.g. extended key usage, certificate extensions) can be designated as either critical or non-critical. The relying party or anyone using the certificate is required to reject it if a critical extension of the certificate is not recognized or cannot be processed. On the other hand, a non-critical extension could be ignored if it is not recognized, but must be processed otherwise. Respecting the extended key usage and extended signature policy extensions, it is up to the particular PKI issuing the certificate its definition as critical or not critical. However, we recommend to designate them as critical in order to counteract attacks explained above.

As users already own one or several key pairs and digital certificates (e.g. software, in smart cards, USB tokens etc.), we also saw the need to design a solution that facilitated them the participation in non-repudiation services according to our thesis proposal but without the need to obtain a new certificate complying with the extension-based solutions proposed herein. In [93], we propose a method to re-issue a subscriber's digital certificate, allowing the incorporation of new key usages or fields, like, for instance, the extended key usage extension or the extended signature policy extension. This method permits to maintain the cryptographic keys while the certificate content is extended or modified according to the particular signature application. Consequently, this proposal may be applied to obtain a new certificate (wrapping the same public key) with extended key usage extension or the extended signature policy extension.

This proposal has the drawback that as long as the source certificate that wraps the same public key is valid, an attacker that compromised the private key could still be capable of producing signatures out of the context of an extended signature policy.

[1]At the time of writing the thesis, OID 2.5.29.65 was not assigned

8.3 Chapter Summary

Signature policies are an important step forward since they customize the requirements an electronic signature must fulfill in a particular transactional context. However, current signature policy definition is focused on the generation and validation rules for a single signature. The management of multiple signatures where relationships exist in a unique transaction is not possible. This scenario limits the usage of signature policies in business scenarios where the presence of more than one signature is a must, like e-commerce, contract signing protocols, e-Government applications, certified email systems or the division principle proposed in Chapter 7.

In this Chapter we have proposed a complete framework to cover this need. In particular, the definition of an extended signature policy along with the generation and validation procedures to be followed by signers and verifiers have been presented. We have designed the solution taking into account current standards. As a result, the extended signature policy framework described herein can be easily integrated into existent signature applications and processes.

The policy definition is given both in ASN.1 and XML, allowing its integration in processes where either ASN.1 or XML signatures have to be generated. Moreover, the framework has been designed to be independent from the particularities of the higher layer protocol or application that uses it. As a result, our design allows a flexible and protocol independent definition of extended signature policies, supporting complex business models.

Chapter 9

An Optimistic Fair Exchange Protocol based on Signature Policies

In this Chapter, we propose a novel fair exchange protocol oriented to Internet transactions where two parties exchange information in a fair and secure manner. The protocol, named OFEPSP+ (improved Optimistic Fair Exchange Protocol based on Signature Policies), is based on the division paradigm and the extended electronic signature policies proposals given in Chapters 7 and 8, respectively.

The Chapter is organized as follows. The protocol scheme is detailed in Section 9.1. Section 9.2 contains some recommendations and guidelines for a practical implementation of the protocol. Finally, the Chapter is concluded in Section 9.3. The basic notation and definitions used along the Chapter are given in the introduction of the thesis (see Chapter 1).

9.1 The Protocol

The objective of the protocol is the fair exchange of origin's message and non repudiation evidence of both origin and receiver. The origin sends a signed message to the receiver while the receiver sends back a proof of receipt of the message. Therefore, both parties are making a commitment in the transaction: the origin cannot repudiate having created the content of the message and having sent it, while the receiver cannot repudiate having received the message. As a fair exchange protocol, the protocol ensures that no party gains an unfair advantage over the other during the protocol execution. Therefore, either both parties obtain the expected information or none of them obtains any useful information from the other. As an optimistic protocol, a Trusted

Third Party (TTP) is included in the design but participating only when a party's misbehavior or protocol error occurs.

Many fair exchange protocols found in literature are designed using symmetric encryption, assuring the undisclosure of the message sent by the origin until the receiver has made a commitment in the transaction [142]. In our case, the protocol, named OFEPSP+, is based on signature policies [71], extended signature policies (see Chapter 8) and the division paradigm (see Chapter 7). The rules that manage the protocol execution and the dispute resolution clauses are specified in the policies. Therefore, the exchange between origin and receiver (and TTP when necessary) totally depends on the policies content.

During the protocol run, several digital signatures are generated by both the origin and the receiver, until valid evidence is produced. Therefore, valid evidence will consist of several digital signatures. According to the extended signature policy rules, each digital signature is not binding on its own, and only when every digital signature, and thus the valid evidence, have been correctly generated, the origin and the receiver are bound to their commitments. The commitment of each party is represented by the group of signatures it has generated. Therefore, valid evidence inherently implies the existence of two commitments, one from each side. As commented above, it is assumed that non-repudiation of origin and non-repudiation of receipt services are being delivered in OFEPSP+. However, it should be noted that different commitments could be established by using the appropriate signature policies, and without having to modify the protocol design.

The signature policy and the extended signature policy references are included as a signed property in each signature performed by the parties, and that will be part of valid evidence. It allows any verifier to ascertain if evidence matches the requirements imposed in the policies. Therefore, signature creation and verification processes can be completely carried out in an automatic and transparent way in accordance with the signature policies rules.

For simplicity and practical purposes, the division principle is only applied to the origin, enhancing the reliability of partial evidence generated by the origin, and that consists of every digital signature generated by him. Therefore, it is being assumed that the origin corresponds to an end user, while the receiver could be represented by an e-commerce Web site. We limit the number of environments to two. As a result, the origin will use two environments, generically named E_1 and E_2, to compose the partial evidence of origin, and verify the evidence being generated during the transaction. Both environments must comply with the models provided in Section 6.2. In addition, the reliability of origin's partial evidence can be easily increased as OFEPSP+ design permits to add new environments in a straightforward manner. It is important to remark that the extended signature policy that manages the transaction permits not only to implement the division principle on the origin's side but also to assure the fairness of the protocol.

OFEPSP+ consists of one main protocol, explained in Section 9.1.3, and two subprotocols, named recovery subprotocol and abort subprotocol, explained in Sections 9.1.4 and 9.1.5 respectively.

The binding between the signature creation data and the corresponding environment must comply with either a well established procedure or the environment attestation technique, as explained in Section 7.2.5. In this Chapter, we follow the environment attestation technique for environments E_1 and E_2. Moreover, the transaction data must not be formatted following complex or rich formats, as explained in Section 7.2.6. Besides, a template is used by the parties in order to fix the information to be sent by the origin. This template is referenced by the template identifier *tpl_id*. The template must be defined by the receiver according to the transaction needs, and the message sent by the origin must be further processed by the receiver taking into account the template information.

Finally, each message exchanged includes a protocol identifier ℓ that permits to uniquely identify the protocol run it belongs to.

9.1.1 Entities of the protocol

Before formally detailing the protocol, the function of each participant entity is briefly explained:

Origin (O). It is the entity that initiates the transaction, and that wants to send a message to the receiver with the corresponding non-repudiation evidence of origin. The origin expects a non-repudiation evidence of receipt from the receiver. This entity owns two different environments, E_1 and E_2, for evidence generation and evidence verification.

Receiver (R) It is the entity that receives a message from a origin and the corresponding non-repudiation evidence of origin. The receiver shall send a non-repudiation evidence of receipt to the origin. This entity owns a single environment for evidence generation and evidence verification.

TTP-SP It is a Trusted Third Party that implements the roles of the Signature Policy Issuer and the Signature Policy Publication Authority, according to [71]. Normally, these roles are held by different entities, but other configurations are possible if desired. The TTP-SP holds both roles in the protocol in order to make the explanation clearer. The TTP-SP has certain signature policies - both simple and extended - configured and available both to the origin and receiver.

TTP It is a Trusted Third Party which participates in the recovery and abort subprotocols. Thus, it acts in optimistic mode, that is, only when an abnormal situation occurs in the main protocol.

9.1.2 Evidence exchanged

Partial and valid evidence produced and verified during the protocol are described next, following the notation given in Chapter 1:

$PNRO_1 = S_{SCD_{E_1}}\left(S_O^{E_1}\left(m, \ell, tpl\_id | SP/extSP\right)\right)$

Attested digital signature acting as first partial non-repudiation evidence of origin of message m, using the environment E_1.

$PNRR_1 = S_R\left(m, \ell, tpl\_id | SP/extSP\right)$

Digital signature acting as first partial non-repudiation evidence of receipt of message m, being a sequential signature respecting $PNRO_1$.

$PNRO_2 = S_{SCD_{E_2}}\left(S_O^{E_2}\left(PNRR_1 | SP/extSP\right)\right)$

Attested digital signature acting as second partial non-repudiation evidence of origin of message m, that countersigns $PNRR_1$ using environment E_2.

$PNRR_2 = S_R\left(PNRO_2 | SP/extSP\right)$

Digital signature acting as second partial non-repudiation evidence of receipt of message m, that countersigns $PNRO_2$.

$NRE = S_{SCD_{E_1}}\left(S_O^{E_1}\left(PNRR_2 | SP/extSP\right)\right)$

Attested digital signature that composes the valid non-repudiation evidence that ties down both the origin respecting the message m sending and content creation (non-repudiation of origin) and the receiver respecting the message m reception (non-repudiation of receipt). It is generated by the origin, countersigning $PNRR_2$ and using environment E_1.

$NRE_{TTP} = S_{TTP}\left(PNRR_2 | SP/extSP\right)$

Digital signature that composes the valid non-repudiation evidence that ties down both the origin respecting the message m sending and content creation (non-repudiation of origin) and the receiver respecting the message m reception (non-repudiation of receipt). It is generated by the TTP when the recovery protocol has been executed, countersigning $PNRR_2$.

As can be seen, every signature is generated (and must be verified) following the procedures specified in signature policies SP and extSP. In addition, $PNRO_1$, $PNRO_2$ and NRE must be attested by the environment where such signatures are generated. Finally, $PNRO_1$ and $PNRR_1$ signatures are generated over the message m, the protocol identifier ℓ and the template identifier *tpl_id*.

9.1.3 Main protocol

In the main protocol, the origin initiates the transaction by sending the signed message to the receiver by means of E_1. Afterwards, the origin uses both E_1 and E_2 to complete the valid evidence, and finalize the transaction. The receiver will exchange several intermediate evidences with both E_1 and E_2, until the final valid evidence is generated. Next, the main protocol is formalized using the notation given in Chapter 1:

(1) $O_{E_1} \leftarrow R : \mathit{tpl}\left[\mathit{tpl\_id}\right], S_R\left(\mathit{tpl}\left[\mathit{tpl\_id}\right]\right)$
First, the origin (O) requests the template identified by *tpl_id* to the receiver (R) by means of E_1 (1).

(2) $O_{E_1} \leftarrow$ *TTP-SP*: $SP, S_{TTP-SP}\left(SP\right), extSP, S_{TTP-SP}\left(extSP\right)$
In step (2) the origin retrieves the signature policy SP and the extended signature policy extSP necessary to communicate with the receiver and generate protocol evidence. Once the origin has obtained the template and the policies, he can produce the message and $PNRO_1$ taking into account this information.

(3) $O_{E_1} \rightarrow R : m, \ell, \mathit{tpl\_id}, PNRO_1$
In step (3), the origin sends the message m, a unique protocol identifier ℓ, the template identifier *tpl_id* and the $PNRO_1$.

(4) $R \leftarrow$ *TTP-SP*: $SP, S_{TTP-SP}\left(SP\right), extSP, S_{TTP-SP}\left(extSP\right)$
(5) $R \rightarrow O_{E_2} : m, \ell, \mathit{tpl\_id}, PNRO_1, PNRR_1$
The receiver retrieves the signature policy and the extended signature policy (4), if not obtained yet, and validates the received $PNRO_1$. Afterwards, the receiver generates and sends the $PNRR_1$ to the origin's environment E_2 (5). The receiver must also send all the information received from the origin in step (3) in order to allow him to validate the initiated transaction using the environment E_2. Step (4) can be avoided for efficiency purposes if the receiver accesses the TTP-SP once and afterwards manages a local copy of the signature policies, provided that they are within their validity periods.

(6) $O_{E_2} \leftarrow$ *TTP-SP*: $SP, S_{TTP-SP}(SP), extSP, S_{TTP-SP}(extSP)$

(7) $O_{E_2} \rightarrow R : PNRO_2$

The origin must generate the $PNRO_2$ (7) only if the information received in step (5) corresponds to a desired transaction and $PNRO_1$ and $PNRR_1$ are correctly verified. For that purpose, the origin must download the signature policy SP and the extended signature policy extSP to the E_2 environment (6).

(8) $R \rightarrow O_{E_1} : PNRR_2$

Once the origin has confirmed the transaction by means of the $PNRO_2$, the receiver sends the $PNRR_2$ to the origin's environment E_1 (8).

(9) $O_{E_1} \rightarrow R : NRE$

In the last step (9) the origin completes the transaction by sending the NRE to the receiver. It should be mentioned that, before generating NRE, the origin must validate $PNRR_2$. To validate $PNRR_2$, the origin must use $PNRO_2$, which is obtained from the signature embedded in $PNRR_2$. To validate $PNRO_2$, the origin must use $PNRR_1$, which is obtained from the signature embedded in $PNRO_2$. Finally, to validate $PNRR_1$, the origin must use m, ℓ and *tpl_id*, know from step (1) (see Section 9.1.2 for detailed information about partial evidence composition).

Therefore, valid evidence NRE is generated using two environments of the origin and a third environment of the receiver. Evidence verification is also supported by the two environments of the origin[1]:

- Partial evidence $PNRO_1$ is generated by the origin in E_1 at step (2) but also verified by the origin itself in E_2 at step (6).
- Partial evidence $PNRR_1$ is generated by the receiver in step (5) and verified by the origin both in E_2 in step (6) and in E_1 at step (9).
- Partial evidence $PNRO_2$ is generated by the origin in E_2 at step (7) but also verified by the origin itself in E_1 at step (9).
- Partial evidence $PNRR_2$ is generated by the receiver at step (8) and verified by the origin only in E_1 at step (9), as it comprises the last step of the protocol.

It is important to remark that the signatures that correspond to the evidences generated in the protocol ($PNRO_1, PNRR_1, PNRO_2, PNRR_2$ and NRE or NRE_{TTP})

[1]Please note that the protocol is designed to enhance the reliability of evidence produced and verified by the origin, not the receiver.

must be generated according to electronic signature standards (see Section 9.2). Thereby, references to the signature policies used in the protocol (both simple and the extended) are included as signed properties in the specific electronic signature format chosen for the transaction (i.e. XAdES, CAdES). This permits any party to know and retrieve the signature policies and avoids an attacker to modify the referenced signature policies.

Although not shown above, evidences $PNRO_1, PNRR_1, PNRO_2, PNRR_2$ and NRE must be time-stamped. The time-stamp is an assertion of proof given by the Time-Stamping Authority (TSA) that the datum existed before the specified time. The time-stamping procedure must be carried out according to known standards [5], and implies the participation of a TSA.

9.1.4 Recovery subprotocol

The recovery subprotocol allows the receiver to obtain evidence NRE in case of a protocol interruption or origin's misbehavior, and must be executed if the receiver does not receive the NRE within a specific time interval. OFEPSP+ recovery subprotocol consists of the next steps:

(1) $R \rightarrow TTP : H\left(m, \ell, tpl\_id\right), \ell, PNRO_1, PNRR_1, PNRO_2, PNRR_2$
if (protocol aborted) then
(2a) $TTP \rightarrow R : S_{TTP}\left(S_O\left(abort, \ell | SP/extSP\right) | SP/extSP\right)$
else
(2b) $TTP \leftarrow$ $TTP\text{-}SP$: $SP, S_{TTP-SP}\left(SP\right), extSP, S_{TTP-SP}\left(extSP\right)$
(3b) $TTP \rightarrow R, O_{E_1}, O_{E_2} : NRE_{TTP}$

In (1) the receiver sends the produced partial evidence $PNRO_1, PNRR_1, PNRO_2$ and $PNRR_2$ to the TTP. Also, and in order to protect the privacy of the parties, the information signed in $PNRO_1$ and $PNRR_1$ - the message, the protocol identifier and the template identifier - is not sent in plain text but the hash of their concatenated values. Yet the TTP is still able to verify $PNRO_1$ and $PNRR_1$ by directly using the hash, provided that a digital signature scheme based on public key cryptography is used (e.g. RSA, DSA, ECDSA). The TTP must decrypt the attested digital signature of $PNRO_1$ and $PNRR_1$ using the corresponding public keys, obtaining the hash of the signed information, which must correspond to the value of $H\left(m, \ell, tpl\_id\right)$. ℓ is sent in (1) to allow the TTP to retrieve and update the information associated to the transaction.

If the protocol has already been aborted, the TTP merely forwards the abort evidence to the receiver (2a). On the other hand, the TTP generates the NRE_{TTP} taking into account the referenced signature policies - (2b) and (3b) -, but only in the first request. Evidence must be stored in a local database along with the received information. Subsequently, the TTP will reuse it, improving the efficiency.

9.1.5 Abort subprotocol

The abort subprotocol allows the origin to abort the protocol execution if a malicious behavior of the receiver is suspected, an error during the protocol run has occurred or, more importantly, a fraudulent transaction is detected during evidence verification at any environment. The abort subprotocol can be executed by the origin at any step of the protocol and from any environment. OFEPSP+ abort subprotocol consists of next steps:

(1) $O_{E_1|E_2} \rightarrow TTP : abort, \ell, S_O\left(abort, \ell | SP/extSP\right)$
if (recovery protocol executed) then
(2a) $TTP \rightarrow O_{E_1|E_2} : NRE_{TTP}$
else
(2b) $TTP \leftarrow$ *TTP-SP*: $SP, S_{TTP-SP}\left(SP\right), extSP, S_{TTP-SP}\left(extSP\right)$
(3b) $TTP \rightarrow O_{E_1|E_2} : S_{TTP}\left(S_O\left(abort, \ell | SP/extSP\right) | SP/extSP\right)$

The abort subprotocol is designed to permit the origin to prevent an attacker or malicious receiver from producing valid evidence NRE or NRE_{TTP}. In this sense, an abort subprotocol execution will not lead to a NRE_{TTP} receipt unless the origin produced $PNRO_2$ at environment E_2 (see step (7) of main protocol). Consequently, the transaction information must have been verified by the origin using both environments.

For efficiency purposes, $S_{TTP}\left(S_O\left(abort, \ell | SP/extSP\right) | SP/extSP\right)$ is only generated in the first time (2b) and (3b), being reused in subsequent executions of the abort subprotocol. On the other hand, if the protocol has been recovered (2a), the TTP just retrieves the NRE_{TTP} from its data base, forwarding it to the origin.

9.1.6 Dispute resolution

An important issue in fair exchange protocols is the dispute resolution, which has to be carried out in two cases:

- If the receiver claims to have received message m while the origin denies having sent it to the receiver.
- If the origin claims to have sent message m while the receiver denies having received it from the origin.

The dispute resolution objective is to settle who is lying and who is saying the truth. For that purpose, evidence collected during the protocol execution must be evaluated by a fair and neutral third party, for instance, a judge. This judge is normally a physical person who, by using the appropriate software, can determine the validity of the evidence.

In this protocol, and in both cases above, the evidence that must be presented to the judge is the NRE, NRE_{TTP} or the abort token countersigned by the TTP. Any presented evidence must have been generated according to protocol requirements and the signature policies rules. The protocol design assures that either both parties obtain the NRE (or NRE_{TTP}) or none of them can gain any advantage over the other. Valid evidence can have been generated either by the origin and receiver (NRE) or by both of them and the TTP (NRE_{TTP}).

Therefore, the judge must verify the correctness of evidence, in the sense of digital certificates used by the parties, the validity of the digital signatures and the rest of the information that should have been included in the evidence (see Section 9.2 for further information), always according to the signature policies statements. The existence and correctness of evidence determines whether the claims of origin or receiver are right or wrong.

Furthermore, OFEPSP+ design allows detecting origin's misbehavior in case the origin aborted the protocol after having received the $PNRR_2$ from the receiver. After receiving $PNRR_2$, the origin is capable of producing valid evidence NRE. Aborting the protocol would imply that the receiver was not able to compose a valid evidence, while the origin did. However, and during the dispute resolution, the receiver would present the abort token, signed by the origin and countersigned by the TTP. Consequently, it is easy to demonstrate that the origin misbehaved.

9.2 Implementation Guidelines

This Section contains guidelines for implementing OFEPSP+ and recommendations about signature formats to use, methods for obtaining the validation information and a signature policy architecture proposal that can be integrated in the protocol design. Also, requirements for the communication channels are provided. Appendix F contains an example of an extended signature policy for OFEPSP+.

The architecture proposed is in accordance with [71], but it is applicable not only to the management of simple signature policies but also to the extended signature policy defined in Chapter 8.

9.2.1 Architecture

Signature policies can be applied in private and public sectors, and in open and closed environments. In any case, the policy is defined by the entity that needs those requirements to be enforced. This entity specifies the technical and procedural requirements for electronic signature creation and validation, in order to meet a particular business need. The entity is called the Signature Policy **Issuer**. The issuer can be a legal person (e.g. an organization) or a natural person (acting under a professional function) that establishes the rules that must be followed by his community of users when generating/validating electronic signatures.

A signature policy must be made available to the transacting parties while it is still valid. The Signature Policy **Publication Authority** is responsible for that, and the publication task must be carried out assuring the reliability of both the process itself and the information published.

Signature Policy Issuers may disappear once they have issued their policies and Policy Publication Authorities are not obliged to maintain the signature policies after they expire. But users may still want to validate electronic signatures created under a signature policy a long time after its expiration date. The task of archiving the signature policies in a publicly accessible repository is done by the Signature Policy **Archiving Authority**.

Next Figure 9.1 shows an architecture where above roles are played by three different independent entities. The issuer entity is an external organization to which the receiver is adhered.

Each entity carries out a specific task in the architecture. One of the key entities is the Signature Policy Publication Authority. This entity allows both origin and receiver access a repository for retrieving the signature policy to be used during the protocol execution. The role of the Signature Policy Issuer is not needed during the protocol execution, but obviously it is necessary before the protocol can take place. The publication task can be carried out by the Issuer as well.

As it can be seen in Figure 9.1, the receiver communicates with the Issuer for selecting, among all available policies, the signature policy to be used during the protocol execution. Issuers are normally linked to receivers, and sometimes the receiver generates its own policy. In a normal scenario it is the receiver the one who imposes the requirements for buying a resource in his (e-)commerce. Obviously, the origin can decide whether to accept or not these requirements, by reviewing the signature policy rules and specific receiver's web site conditions.

After that, the receiver has to make the signature policy available to origins. For that purpose the signature policy is uploaded to the Publication Authority repository.

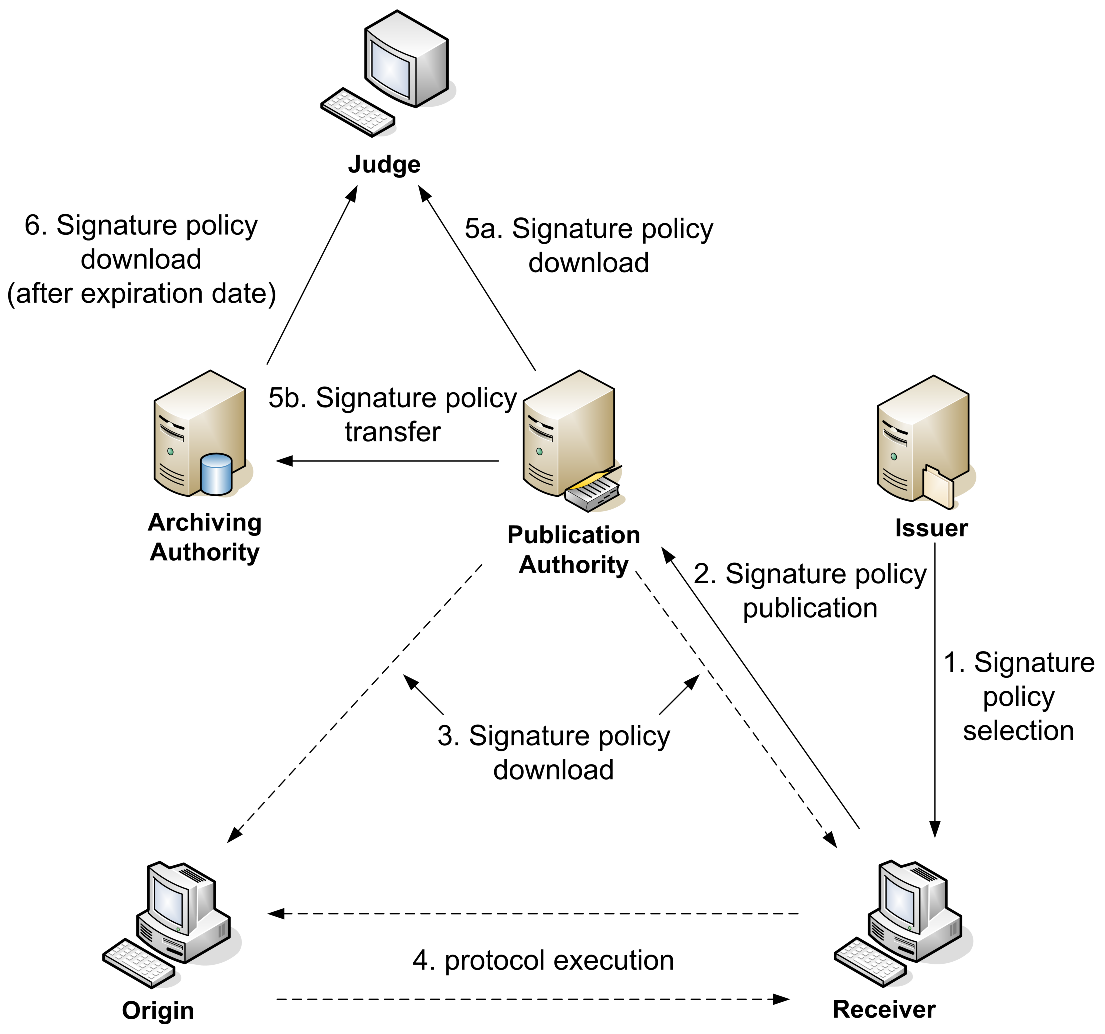
Judge
6. Signature policy download (after expiration date)
5a. Signature policy download
5b. Signature policy transfer
Archiving Authority
Publication Authority
Issuer
2. Signature policy publication
1. Signature policy selection
3. Signature policy download
4. protocol execution
Origin
Receiver

Figure 9.1: Architecture of a signature policy scenario.

Once origins and receivers can obtain the signature policy, the fair exchange protocol can be launched.

It is important to remark that while the signature policy is still valid, the parties can retrieve the signature policy from the repository located at the Publication Authority in order to create and validate the signatures. Once the signature policy has expired, the Publication Authority should forward it to the Archiving Authority, allowing the signatures to be validated beyond the end of the validity of the signature policy (e.g. a judge has to resolve a dispute). However, no more signatures should be created under the rules of an expired signature policy.

9.2.2 Requirements for the communication channels

The architecture proposed in Figure 9.1 implies several message transmissions among involved parties. This Section sets out the requirements for the communication channels.

Communication channels are usually categorized as follows [214]:

Unreliable channel. A channel that may not deliver messages randomly.

Resilient channel. A channel that reliably delivers any message to the other end, after some finite but unknown amount of time.

Reliable channel. A channel that delivers any message to the other end after a fixed and known delay.

Next, all communication channels in the architecture are analyzed using previous classification and the minimum requirements of each communication channel are discussed[1]:

- *Receiver* ← *Issuer*

  This channel is used by the receiver for downloading the signature policies files. Because the receiver will make as many attempts as necessary for successfully downloading the files, even an unreliable communication channel could be used.

- *Origin* ← *Publication Authority*

  This channel may be needed only once in case the simple signature policy to use is the same for the rest of signatures. In that case, the origin would access the Publication Authority only once when performing the $PNRO_1$, to obtain the signature policies needed for the rest of the protocol. From that point onward, the origin would re-use the downloaded policy for partial evidence validation and generation. The extended signature policy must be the same along the whole process, and as such it will only be needed to be downloaded once. As it is assumed that the origin will try to download the signature policies until successfully obtained, a reliable, resilient or unreliable communication channel could be used.

  On the contrary, if several simple signature policies are used during the protocol, the origin will have to access the Publication Authority each time a new policy is referenced in a signature being validated, or a new policy is to be used for the generation of certain signature. Once the protocol is started, delays at message receptions may be taken into account by the parties. In order to avoid that a long delay causes an abort or recovery of the protocol, it is recommended to use a reliable communication between the origin and the Publication Authority. If this requirement cannot be fulfilled, then a resilient channel could be used but the parties are advised to be more tolerant respecting response times. It should be noted that the origin could download every simple signature policy referenced from the extended policy at first step, avoiding this situation.

[1]When referring to origin, it includes both E_1 and E_2 environments

- *Receiver* ← *Publication Authority*
  In the same manner as above, it is recommended to use a reliable communication between the receiver and the Publication Authority, in case the required signature policies have not been locally stored after the signature policy selection (step 1 in Figure 9.1), or the simple signature policy to use is not the same along the whole protocol. If this requirement cannot be fulfilled, then a resilient channel could be used but the parties are advised to be more tolerant respecting response times.

- *Origin* ↔ *Receiver*
  During the protocol execution, communication errors that could cause messages losses might make the abort or recovery subprotocols be executed. Same justification as previous point applies here.

- *Publication Authority* → *Archiving Authority*
  A signature policy transference from the Publication Authority to the Archiving Authority does not require a communication channel with a high level of reliability. An unreliable communication channel suffices, provided that the entities will retry the transmission until it successes. It should be mentioned that, if an error occurs during the transmission, the Publication Authority must not delete the file from its repository. Otherwise, no electronic signature generated under this signature policy could be validated anymore.

- *Judge* ← *Publication Authority and Archiving Authority*
  Finally, this case can be argued to be exactly the same as in previous point, so it is supposed that the judge will make as many attempts as necessary to successfully download the signature policy.

Next table 9.1 summarizes minimum requirements for existing communication channels:

9.2.3 Electronic signature format

Section 3.2 mentioned that Advanced electronic signature (AdES) formats that include the signature policy reference and a time reference are suffice to fulfill the requirements imposed by ISO model on non-repudiation for generic non-repudiation tokens (GNRT) and non-repudiation tokens specific to the service being provided. However, if the transaction context requires the long-term assurance of non repudiation of the actions performed, as it may be the case in the proposed protocol, it is of utmost importance to incorporate additional information to the electronic signature and fulfill certain requirements during its generation.

Communication channel	Minimum requirement
$Receiver \leftarrow Issuer$	Unreliable
$Origin \leftarrow Publication\ Authority$	
single access	Unreliable
accesses during the protocol	Resilient
	Reliable (recommended)
$Receiver \leftarrow Publication\ Authority$	
single access	Unreliable
accesses during the protocol	Resilient
	Reliable (recommended)
$Origin \leftrightarrow Receiver$	Resilient
	Reliable (recommended)
$Publication\ Authority \rightarrow Archiving\ Authority$	Unreliable
$Judge \leftarrow Publication\ Authority\ /\ Archiving\ Authority$	Unreliable

Table 9.1: Minimum requirements for communication channels in the protocol architecture.

The main aim is to provide electronic signatures with all necessary information that allows them to be successfully verified in the future, even if a long time has elapsed since their creation. The reason is that in certain scenarios, electronic transactions relate to contractual operations. It is possible that the commitments made in the transaction should still be valid long time after the transaction took place (e.g. dispute resolution).

As mentioned above, signature policies have been defined both in ASN.1 [73] and XML [70]. This allows their use in CAdES and XAdES formats, respectively. Incorporating a signature policy (or a reference to it) in an AdES upgrades it to an AdES-EPES format (Explicit Policy-based Electronic Signature), a format from which AdES-T, AdES-C, AdES-X or AdES-A can also be built. In particular, we suggest an AdES-EPES format (either in XML or ASN.1) that contains the following validation information:

- Timestamp over the digital signature value.
- Certification path.
- Certificates revocation status information.

Time stamping a digital signature provides evidence that the signature has been created before the time of stamping, and requires the presence of a Time-stamping Authority (TSA). Certification path implies capturing all the certificates from the certification path, starting with that from the signer and ending up with the self-signed certificate from one trusted root. Thereby, the verifier can ascertain that the certification path was valid according to naming or certificate policies constraints. Finally, revocation status information of all certificates presented in the signature completes it with the necessary information for assuring its validity in a future, although this information could not be obtained anymore. Revocation information can be obtained by accessing an Online Certificate Status Protocol (OCSP) service, if available, or by retrieving the Certificate Revocation List (CRL) otherwise.

Validation information above provides the electronic signature with long term proof. By combining time reference with certification path and revocation status information, verifiers can be sure, at any time (even after certificate expiration or revocation), about the validity of the signature and signer's certificate at the moment of signature generation. Note that a Certification Authority normally deletes a certificate revocation information entry from the CRL as soon as the certificate expires. Moreover, the AdES-EPES allows the generation of electronic signatures according to a signature policy, complying with the conditions imposed by the optimistic fair exchange protocol proposed in this Chapter.

9.2.4 Addition of validation information

Previous information may be collected and added to the signature by the signer or verifiers, depending on the context and particular technological limitations. For instance, if the signer is an individual buyer with limited computer capability, then it is preferable to move the validation information retrieval from the buyer to the seller side.

In case the transaction is performed in a B2B context, assuming that both sides are able to access external systems and none of them have network bottlenecks, the solution is not fixed either. The protocol proposed in this Chapter sets that each side must obtain a time reference for the signatures they generate. Thereby, because the time stamp is obtained before sending the electronic signature, a more accurate time reference is applied by avoiding communication delays. However, it is also possible to establish that each one must obtain the validation information of the other side, in order to be sure that, for example, the revocation status information corresponds to the validation time reference and therefore it is not being obtained long time after the timestamping. Notice that a grace period should be taken into account for allowing revocation requests being processed by the Certification Authorities before the verifier collects the revocation information. If not, the verifier will not obtain reliable information if a revocation request was issued by the signer just before the signature was computed and time-stamped [74, 75].

The solution that fits better with the protocol design is a hybrid one. On the one hand, each side obtains a time stamp over the digital signature it has just generated and adds it to the AdES-EPES, building an AdES-T. On the other hand, the other side must collect the remaining validation information after the AdES-T has been validated. As an example, in step (5) of main protocol the receiver must verify the $PNRO_1$ with time stamp, and then, if successfully verified, collect the certification chain and the revocation status information of all included certificates. Likewise, in step (7) the origin must do the same with the time stamped $PNRR_1$ and receiver's related certificates.

This solution improves the accuracy of the validation information while preserving the closest time reference of each signature. If the communication between origin and receiver has to be as interactive as possible, then no grace period should be applied. Trade-off between accuracy and time-response must be made by the system designer. The way the validation information (certification path and revocation status information) is added to the AdES-T differs, and depends on specific implementation conditions. There are mainly two possibilities: incorporating the validation information itself (AdES-X) or incorporating a reference to it (AdES-C). The former solution allows the AdES being completely independent but of greater size, while the latter minimizes the size of the resulting signature but obliges to store the information in an accessible repository. Due to the need of specific applications for storing referenced information in an AdES-C solution, in a B2C context an AdES-X solution is probably a better choice. In B2B both solutions could be applied.

9.2.5 Addition of new environments

OFEPSP+ design obliges the origin to use two different environment. The reason for the restriction in the number of environments stems from practical issues. A configuration of two environments can be easily managed by end users, like, for instance, using the personal computer as E_1 and a mobile device with cryptographic capabilities as E_2.

However, adding new environments could be of interest if the situation recommends it and it is feasible from technical and usability viewpoints. In particular, incorporating a third environment E_3 is trivial, and would not imply any substantial modification in the protocol. The receiver would only have to send $PNRR_2$ to E_3 in step (8) of main protocol. The origin would then use such environment in step (9) to verify the information and complete the transaction. It should be noted that, in this case, the receiver would have to send m, ℓ, *tpl_id* and $PNRO_1$ as well to allow the complete verification of $PNRR_2$ and $PNRO_1$.

Next, OFEPSP+ main protocol is redesigned to permit the addition of as many environments as desired. Steps (1) to (7) remain the same as in main protocol above:

(1) $O_{E_1} \leftarrow R : tpl\,[tpl\_id]\,, S_R\,(tpl\,[tpl\_id])$

(2) $O_{E_1} \leftarrow$ *TTP-SP*: $SP, S_{TTP-SP}\,(SP)\,, extSP, S_{TTP-SP}\,(extSP)$

(3) $O_{E_1} \rightarrow R : m, \ell, tpl\_id, PNRO_1$

(4) $R \leftarrow$ *TTP-SP*: $SP, S_{TTP-SP}\,(SP)\,, extSP, S_{TTP-SP}\,(extSP)$

(5) $R \rightarrow O_{E_2} : m, \ell, tpl\_id, PNRO_1, PNRR_1$

(6) $O_{E_2} \leftarrow$ *TTP-SP*: $SP, S_{TTP-SP}\,(SP)\,, extSP, S_{TTP-SP}\,(extSP)$

(7) $O_{E_2} \rightarrow R : PNRO_2$

(8) $R \rightarrow O_{E_3} : m, \ell, tpl\_id, PNRO_1, PNRR_2$

(9) $O_{E_3} \leftarrow$ *TTP-SP*: $SP, S_{TTP-SP}\,(SP)\,, extSP, S_{TTP-SP}\,(extSP)$

(10) $O_{E_3} \rightarrow R : PNRO_3$

(11) $R \rightarrow O_{E_4} : m, \ell, tpl\_id, PNRO_1, PNRR_3$

(12) $O_{E_4} \leftarrow$ *TTP-SP*: $SP, S_{TTP-SP}\,(SP)\,, extSP, S_{TTP-SP}\,(extSP)$

(13) $O_{E_4} \rightarrow R : PNRO_4$

...

(k) $R \rightarrow O_{E_n} : m, \ell, tpl\_id, PNRO_1, PNRR_{n-1}$

(k+1) $O_{E_n} \leftarrow$ *TTP-SP*: $SP, S_{TTP-SP}\,(SP)\,, extSP, S_{TTP-SP}\,(extSP)$

(k+2) $O_{E_n} \rightarrow R : NRE$

Being

$PNRO_i = S_{SCD_{E_i}}\left(S_O^{E_i}\,(PNRR_{i-1}|SP/extSP)\right), \forall\ i = 2\ to\ n$

$PNRR_i = S_R\,(PNRO_i|SP/extSP), \forall\ i = 2\ to\ n-1$

$NRE = PNRO_n$

9.3 Chapter Summary

Electronic transactions are prone to generate situations where some users are at a disadvantage to others. Particularly, this Chapter has analyzed the situation where an origin, after sending his sensitive information together with the evidence of origin to the receiver, expects to receive the corresponding evidence of receipt. There is a moment when the receiver has all necessary evidence from the buyer but without having made any type of commitment. For resolving this unfair situation, different protocols, known as fair exchange protocols, have been proposed so far, during which neither origin nor receiver can gain any advantage during the transaction. In case that a dispute arises about what actually happened during the transaction, evidence must be provided to a judge.

In this Chapter a completely new and innovative fair exchange protocol has been proposed. The protocol design is based on the origin's signature environment division

(Chapter 7) and extended signature policies (Chapter 8). Also, a design based on an offline TTP improves the overall performance provided that no misbehavior occurs.

By using this approach, the origin can decide whether trusting or not in the entity that issues the signature policy and if accepts or not the terms established in it. This new contribution in fair exchange protocols allows increasing the confidence in e-commerce, because end users are now an active player that know and evaluate the conditions that will manage the electronic transaction.

On the other hand, compliance to European and International electronic signature standards assures that a solution based on this protocol will be interoperable with other standard e-commerce frameworks, and can be quickly implemented. For easing implementation processes, general guidelines covering key factors have also been widely explained. IT staff that wants to put this protocol proposal in practice must take these factors into account if they want to assure an efficient implementation of the protocol.

Part IV

Evaluation and Conclusions

Chapter 10

Evaluation

This Chapter comprises the evaluation of the thesis contributions, covering the taxonomy (Chapter 6), the paradigm of the division of the signature environment (Chapter 7), the extended electronic signature policy (Chapter 8) and the OFEPSP+ protocol (Chapter 9).

In the first instance, the taxonomy is evaluated in Section 10.1 against the set of general requirements a taxonomy should fulfill. This Section also includes the analysis of the intensive survey and classification of attacks on digital signatures performed, which results are given in Appendix B.

In Section 10.2, the formal proofs that demonstrate the benefits of the division paradigm are given. Also, the proposal presented in this thesis to enhance the signature reliability is evaluated against the attack categories identified in the taxonomy, analyzing to what extent our proposal improves the current state of the art.

The experimental implementation developed is explained in Section 10.3. The implementation serves two purposes. On the one hand, to prove the correctness of the validation algorithm of the extended electronic signature policy framework under several test cases. This algorithm, which pseudo-code is given in Appendix C, is independently evaluated, no matter which protocol would further use it. On the other hand, the implementation intends to prove the executability and feasibility of OFEPSP+ by a simulation of the protocol, and by which the correctness of the division paradigm and the validation algorithm are also proved.

Section 10.4 contains the formal analysis of the security of OFEPSP+ respecting the intruder model of Dolev and Yao, using two tools that implement automated reasoning techniques: the Automated Validation of Internet Security Protocols and Applications (AVISPA) and the Security Protocol ANimator for AVISPA (SPAN). An informal analysis of the fairness property of the protocol is also provided.

10.1 Evaluation of the Taxonomy

A taxonomy should be designed to satisfy a set of general requirements [158]. The evaluation of the taxonomy proposed in Chapter 6 is given in Section 10.1.1. In addition, the results and analysis of an intensive survey and classification of 112 attacks found in the literature (see Appendix B) is presented in Section 10.1.2. This survey intends not only to demonstrate the completeness of the taxonomy but also to review the most relevant attacks on digital signatures along with possible countermeasures.

10.1.1 Evaluation against general requirements

A taxonomy should be **generally acceptable** in the field of application for which it is designed. Obviously, this property can be satisfied only if the taxonomy is accessible by others and approved as valid after some time of study. The taxonomy proposed in this thesis builds on previous work that has had relevant impact in the scientific community. The taxonomy follows the well-known concept of dimension, which has been proved to be a good way for providing a holistic view of the field of study. Though it is still to be seen if the proposed taxonomy is accepted by the community, we are confident of it.

A taxonomy should be **exhaustive** in the sense that it covers all known related specimen. This property is hard to be fulfilled, since the classification of every known phenomenon is near impossible, specially in such a dynamic field like the information technology. However, the evaluation of a taxonomy against real samples is paramount to verify its correctness and completeness. The larger the number of samples classified, the higher the level of assurance. In our case, we have successfully classified 112 attacks (see Appendix B). Moreover, our method of classification permits the taxonomy to evolve along the time due to the refinement stage. Consequently, it can incorporate new categories if required.

A taxonomy should be **mutually exclusive**. Each specimen should be classified under, at the most, one category of the taxonomy. The method of classification provided in Chapter 6 and the design of the taxonomy assure that an attack cannot be classified into multiple categories in a dimension. The possibility to select several subcategories in dimension *Target of the Attack* does not violate this principle, but allows to classify several elements affected by the attack, if necessary.

A taxonomy should be **comprehensible** in a manner that it should be understandable and applicable by non-expert users. On the contrary, our taxonomy requires specific IT security knowledge, requiring the person in charge of the classification to have a deep understanding of security and the attack itself.

A taxonomy should be **deterministic** and **repeatable**. The method applied for the classification should be clear and unambiguous, and it should be possible to repeat the classification of a specimen, obtaining the same result as in previous classifications of the same specimen. In this work, a simple but effective method of classification is provided along with the taxonomy, facilitating a trained user the classification task. However, we do not guarantee that our method of classification is deterministic, though we hope that it can lead to homogeneous classifications when the available information of the attack is detailed enough.

A taxonomy should **use widely accepted terminology** and be **appropriate**. The terms and definitions used by the taxonomy should comply with established and well-known terminology, and it should be based on a reference model and a well-defined set of restrictions (if any) [9]. The proposed taxonomy is based on standard system models [46, 47] and a well-defined threat model, using terms extracted from widely accepted and standard sources. The provided reference model assures that the person in charge of classifying or searching for an attack can know exactly which is the underlying model of applicability.

A taxonomy should be **focused** in order to be useful, being specific to a certain field of knowledge. This taxonomy is particularly focused on attacks on digital signatures, and more specifically on those that may affect the security of the signing and verification processes, which are the most critical stages in the digital signature life-cycle.

Finally, a taxonomy should be **useful** for the users belonging to the field of application. We humbly think that this taxonomy fills a current gap in the field of digital signatures, once their relevance and importance have become obvious after the approval of specific legislation and standards, the spread of related technology and their common application in real-life online scenarios. This systematic categorization of attacks on digital signatures will allow developers to build more robust and secure solutions, counteracting current attacks by designing countermeasures of general applicability.

10.1.2 Survey and classification of attacks on digital signatures

An intensive survey and classification of 112 attacks on digital signatures found in the literature (a few of them proposed by the author in this thesis) has been made, and its results are included in Appendix B. We have found a significant higher number of attacks involved in the signature creation process than attacks intended to subvert the verification process. In particular, the survey covers 81 attacks focused on the signature generation stage, while the remaining 31 attacks correspond to attacks on the verification stage.

It should be mentioned that the survey of attacks does not intend to demonstrate a statistical distribution of the types of attacks on digital signatures. The attacks have been selected from the literature according to their relevance. As a result, no strong conclusion should be made on the likelihood of occurrence of each type of attack.

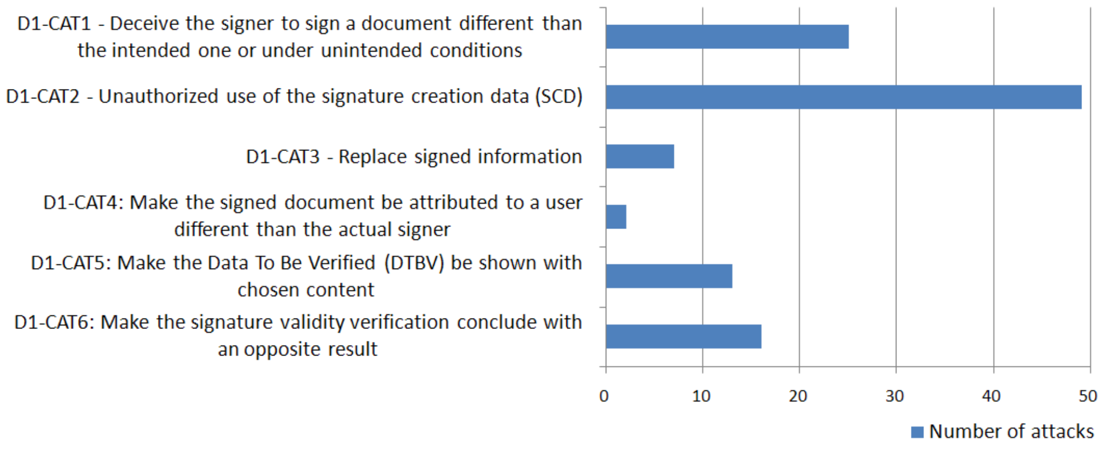

Figure 10.1: Number of attacks per goal category.

Notwithstanding, we do think that some conclusions can be made respecting the impact, dangerousness and profile of the surveyed attacks. Furthermore, the large number of surveyed attacks permits to prove the completeness of the taxonomy to a large extent, as commented in Section 10.1.1.

Figure 10.1 depicts the number of attacks per goal category. It is clear that threats to the signature generation process (represented by categories D1-CAT1, D1-CAT2 and D1-CAT3), and in particular those pursuing goals D1-CAT1 (25 out of 112 (22,3%)) and D1-CAT2 (49 out of 112 (43,7%)), are the most attractive ones for both attackers and researchers. In our opinion, the justification lies in that the generation process is the most critical stage during the life-cycle of a signature, and also the one that is most profitable for the attacker if compromised. In this sense, we see that most of the attacks (almost half of the total) are designed to use the signature creation data for malicious purposes (goal D1-CAT2), followed by attacks aimed at deceiving the user during the signing process (goal D1-CAT1). Few attacks pursued goal D1-CAT3 (7 out of 112 (6,2%)). Observing the attacks on the verification stage, we found few attacks - only 2 - oriented to trick the verifier respecting the identity of the signer, represented by goal D1-CAT4 (2 out of 112 (1,7%)), while the number of attacks according to goal dimensions D1-CAT5 (13 out of 112 (11,6%)) and D1-CAT6 (16 out of 112 (14,2%)) is more balanced.

We consider that it is important to analyze the distribution of attack categories in two cases: the number of attacks that focused on each target versus the goal dimension, and the number of attacks that employed each method of attack versus the goal dimension. These two viewpoints will permit us to discover the targets and methods involved in the most relevant attacks on digital signatures found.

Figure 10.2 shows that the most commonly affected targets, at the generation stage (goals D1-CAT1, D1-CAT2 and D1-CAT3), are: the SCA (15 attacks), the SSCDev

(14 attacks), the SCDev (12 attacks), the Document processor (10 attacks), and the Document to be signed (10 attacks). On the other hand, most commonly affected targets, from the verification viewpoint, are: the SVA (13 attacks) and the Document processor (10 attacks).

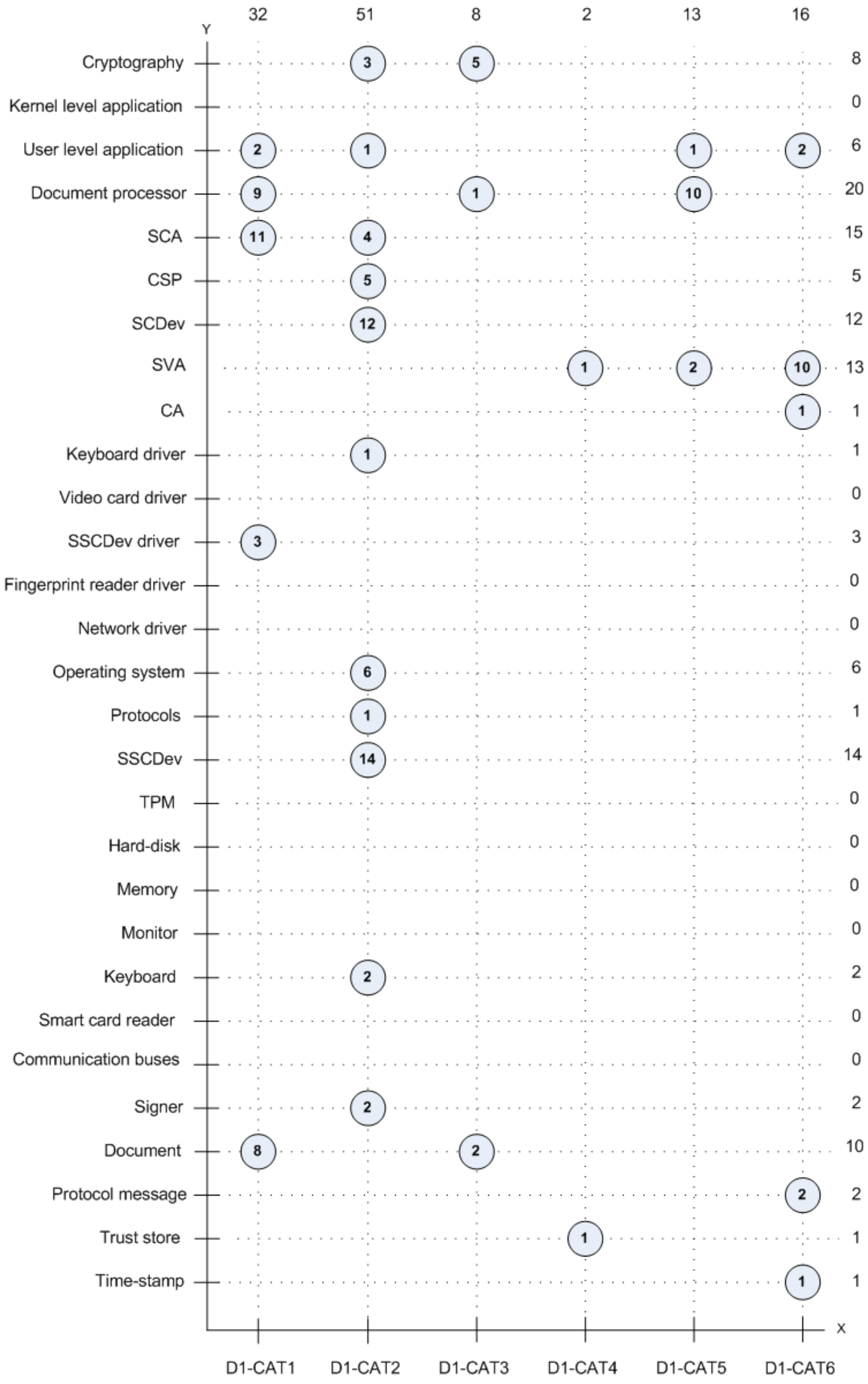

Figure 10.2: Distribution of attacks: Target versus Goal.

These elements are directly involved during the signing and verification operations. Therefore, it is reasonable to think that they are more likely to be attacked than other system components. Consequently, these elements should be carefully designed and implemented to increase the level of assurance of their correctness and trustworthiness. Notwithstanding, the existence of vulnerabilities or weaknesses in other components, like the underlying operating system (6 attacks) or the cryptography used (8 attacks), may open the door for an attack to succeed, no matter the high reliability of the aforementioned elements.

Targets with zero mappings mean that they were not found in the surveyed attacks. However, they can also be a potential victim in an attack on digital signatures. Their direct or indirect participation during a signing operation make them an objective as well.

On the other hand, we present the distribution of methods of attack versus the goal in the cases of the generation and verification phases. Figure 10.3 shows that the distribution of methods of attack applicable to the generation stage versus the goal dimension is homogeneous, though the collection of side-channel attacks (22 attacks in total), content modification methods (10 attacks) and authentication bypass (9 attacks) prevail. In our opinion, this homogeneous distribution proves that there is a wide variety of attack methods that can undermine the security of a signing operation. Also, the distribution respecting axis Y of the Figure demonstrates the specificity of the attacks, each of which is employed to achieve a single specific goal.

Finally, Figure 10.4 illustrates the distribution of methods of attack applicable to the verification stage versus the goal dimension. As in the previous distribution, it can be seen that there is a clear specificity in the attacks surveyed. Also, two methods prevail: those focused on modifying the appearance of the signed document by means of document masquerading attacks (10 attacks), and, to a lesser extent, methods that masquerade the verification result shown (5 attacks).

As can be seen in Figures 10.3 and 10.4, the method of attack raises accurate information about the pursued goal, and vice versa. Each classified method is mapped to just one goal, contrary to the target distribution, where some targets are mapped to more than one goal. This can help making informed decisions when implementing security measures to counteract certain types of attacks or avoid the attacker to achieve a certain goal. As mentioned before, attacks that pursue goal D1-CAT2 are the most dangerous ones, specially those that compromise the signature creation data. Consequently, systems should be designed and implemented to particularly mitigate the risks associated to attacks that entail goal D1-CAT2.

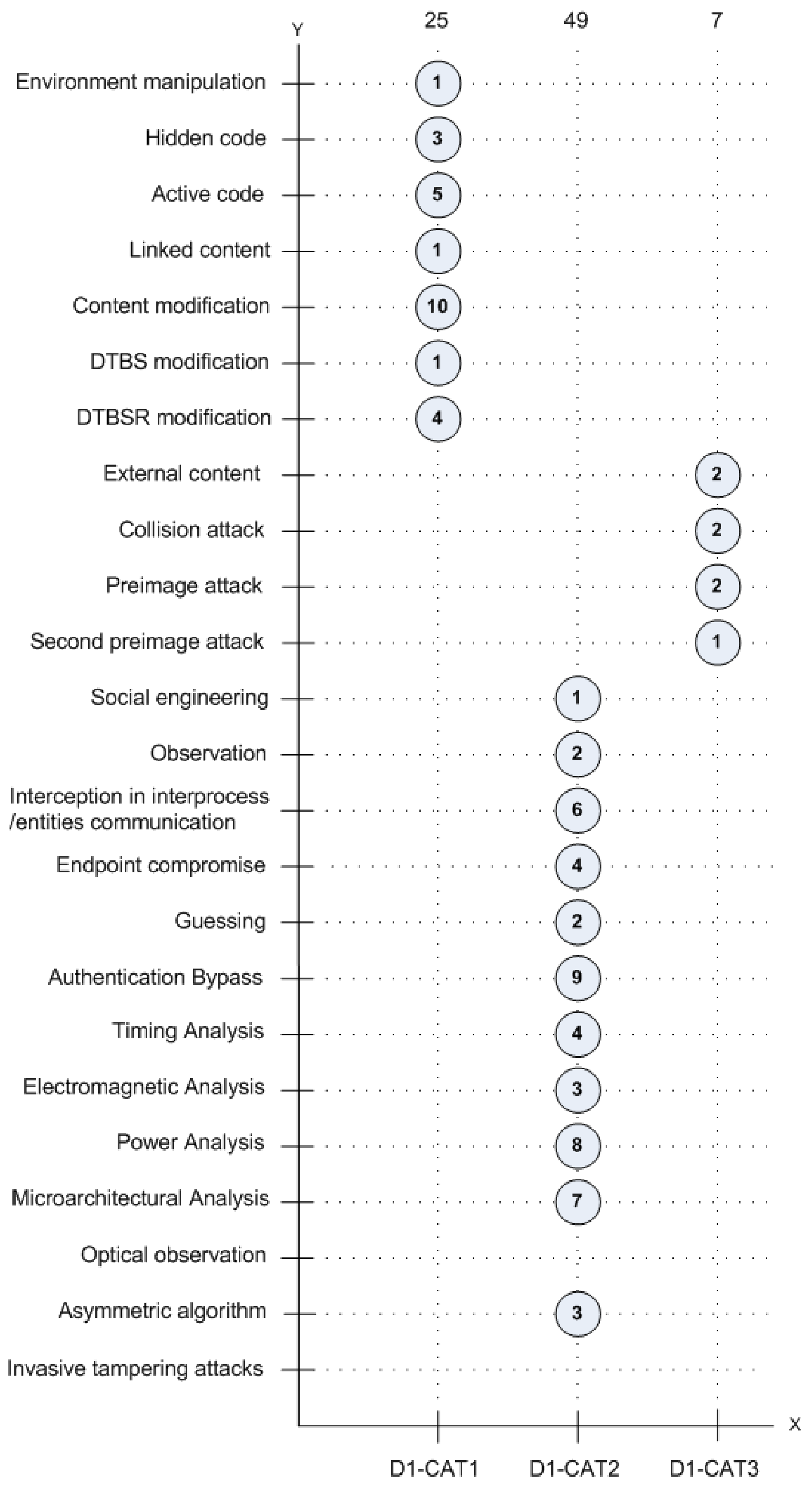

Figure 10.3: Distribution of attacks: Method versus Goal (generation).

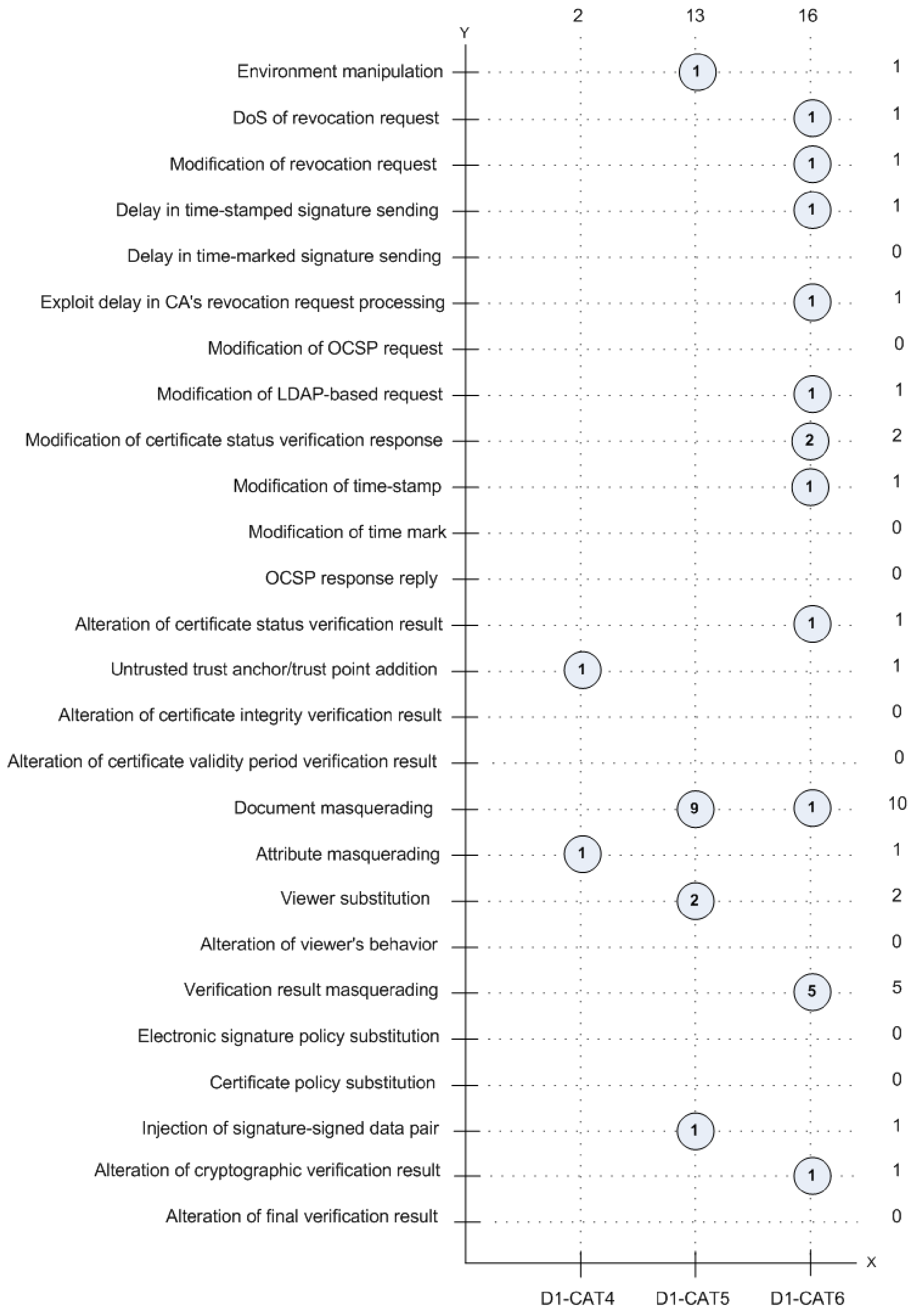

Figure 10.4: Distribution of attacks: Method versus Goal (verification).

10.2 Analysis of the Enhancement of Evidence Reliability

In this Section we evaluate to what extent our proposal enhances the reliability of digital signature-based evidence, demonstrating the improvement achieved. In the first instance, the formal proofs that demonstrate the benefits of the division of the signature environment are given in Section 10.2.1. The demonstration also raises some recommendations on the configuration of the set of environments, gives some numerical examples and outlines a theoretical approach to achieve perfect security. In the second instance, Section 10.2.2 provides a theoretical analysis of the reduction achieved by our thesis proposal on the probability of a successful attack for each attack method identified in the taxonomy of Chapter 6. In this case, the thesis proposal has a broader concept, including not only the division paradigm but also the extended electronic signature policies, and associated contributions.

10.2.1 Formal proofs

The formal proofs that demonstrate the benefits of the division paradigm are given in this Section. Definitions 1 (evidence establishment), 2 (attack on an environment) and 3 (probability of a successful attack (PSA)) given in Chapter 7 apply here.

10.2.1.1 Provable benefits of using several environments

Next, the proofs of the benefits of using several environments for evidence establishment are given.

Consider the scenario shown in Figure 10.5. In the Figure, a division of the environment is represented. As can be seen, the user owns several and different environments for the evidence establishment. As mentioned above, evidence establishment may imply a signature creation or a signature verification.

From this scenario we establish the next formal rationale.

Suppose a set of environments $Set(E)$ of size $n \geq 2$, being n the number of possible environments available to the user, each of which with a specific PSA. The PSA on $Set(E)$ is given by next equation:

$$PSA\left(Set\left(E\right)\right) = \prod_{i=1}^{n} PSA\left(E_i\right) \tag{10.1}$$

It should be noted that we are considering the resultant PSA as the probability of occurrence of n independent events. However, subverting the security of a process in which several environments are needed implies a kind of collaborative attack from the attacker's side. Considering independent events means that what is being considered

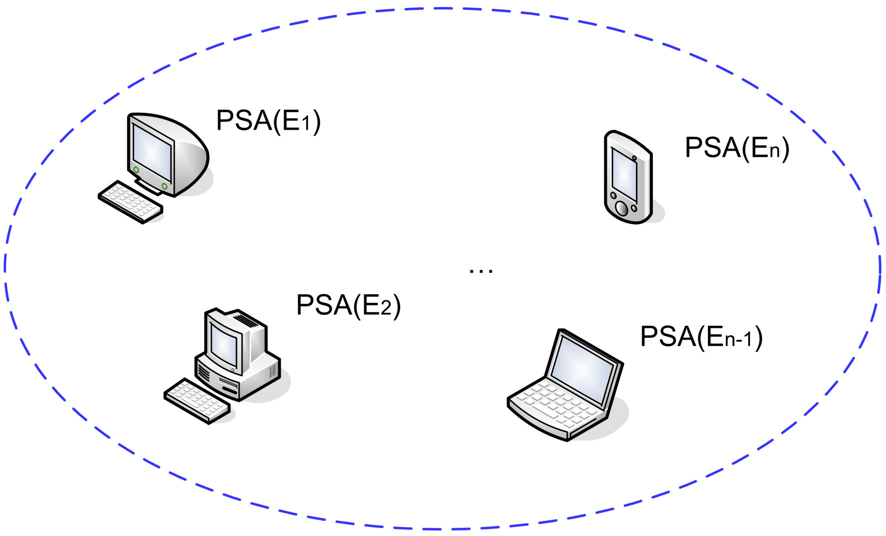

Figure 10.5: A division of the signature environment.

to calculate the resultant PSA for a set of environments is only the PSA on each environment. Therefore, it is not considered whether the attacks carried out on E_1 and E_2 are collaborative, and thus are capable of undermining the division principle, or, on the contrary, are independent attacks carried out by independent malicious agents that do not share their achievements. In the latter, though both environments were compromised, there would be no fraudulent valid evidence that consisted of the signatures of both environments. Each attacker would have generated a fraudulent signature on his own, but not shared it with the other one to compose the valid evidence. Consequently, an attack on the division principle would have not succeeded. Thereby, the actual PSA for a set of environments is even lower, as more a specialized attack is required. Notwithstanding, we will maintain this value of PSA for the analysis.

Claim 1. Increasing the number of environments needed in conjunction to establish evidence enhances the reliability of the evidence establishment, and, as a result, the evidence itself.

Proof 1. Let $PSA(E)$ be the probability of a successful attack on a single environment E. The PSA of $Set(E)$ is always lower than the PSA of a single environment E if an environment E' member of the set $Set(E)$ has a PSA lower than or equal to the PSA of the environment E, and at least one of the rest of the environments members of the set has a PSA lower than 1.

$$\begin{gathered} PSA\left(Set\left(E\right)\right) \;<\; PSA\left(E\right),\; if\; \exists\; E' \;\in\; Set\left(E\right) \;/\\ PSA(E') \leq PSA(E) \;\wedge\; \prod_{i=1}^{n-1} PSA\left(E_i\right) \neq 1, E_i \in Set\left(E\right) \;\wedge\; E_i \neq E' \end{gathered} \tag{10.2}$$

Remarks. The direct consequence of *Proof 1* is that adding new environments will always improve the security of the system by decreasing the final PSA. The assumption of adding environments to the set with a PSA lower than 1 is reasonable, as the user would never use an environment which is known a priori to be compromised.

10.2.1.2 Provable benefits of using heterogeneous environments

Section 10.2.1.1 has proved that using several environments increases the level of security of the system. This Section analyses the impact of configuring a set $Set(E)$ of environments to be used for evidence establishment in case the set consists of either homogeneous or heterogeneous environments.

Definition 1. We define *homogeneous environments* as those environments that can be attacked by the same type of attacker carrying out the same attack process. That is, their implemented security measures, configuration and type of potential attacker are the same. As a result, the PSA for those environments remains the same.

Let $PSA_{hom}(E)$ be the resultant PSA of n homogeneous environments:

$$\begin{aligned} PSA_{hom}(E) &= PSA(E).PSA(E)...PSA(E) \\ PSA_{hom}(E) &= \prod_{i=1}^{n} PSA(E) = (PSA(E))^{n} \end{aligned} \tag{10.3}$$

Figure 10.6 depicts a scenario where the user owns homogeneous environments.

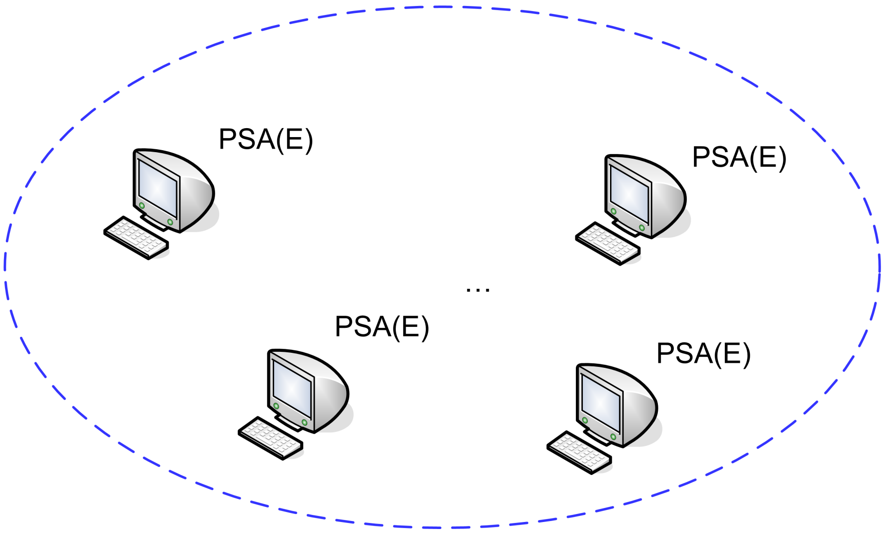

Figure 10.6: A homogeneous set of environments.

Definition 2. We define *heterogeneous environments* as those of different nature, different implemented security measures, configuration and/or different type of potential attacker. As a result, heterogeneous environments can have the same or different PSA.

Let $PSA_{het}(E)$ be the resultant PSA of n heterogeneous environments:

$$\begin{aligned} PSA_{het}(E) &= PSA(E_1) . PSA(E_2) ... PSA(E_n) \\ PSA_{het}(E) &= \prod_{i=1}^{n} PSA(E_i) \end{aligned} \tag{10.4}$$

From the definitions above, we establish the next formal rationale.

Claim 2. Replicating the same environment in the set of environments $Set(E)$ (homogeneous environments) always provides a higher level of security than a configuration based on heterogeneous environments providing that the chosen environment is the most secure one among all possible environments.

Proof 2. From 10.3 and 10.4 we can deduce that:

$$PSA_{hom}(E) < PSA_{het}(E) \Rightarrow (PSA(E))^n < \prod_{i=1}^{n} PSA(E_i), \tag{10.5}$$

$$if \ \exists \ PSA(E_j) < PSA(E_i), \ \forall \ i = 1...n \ and \ j \in \{1...n\} \tag{10.6}$$

Claim 3. In a more general manner, replicating the same environment in the set of environments $Set(E)$ (homogeneous environments) provides a higher level of security if the resultant PSA in 10.3 is lower than that obtained from a configuration based on heterogeneous environments 10.4.

Proof 3. Based on $PSA_{hom}(E)$ and $PSA_{het}(E)$ given in 10.3 and 10.4 respectively, there can be a configuration of homogeneous environments where:

$$PSA_{hom}(E_j) < PSA_{het}(E) \Rightarrow (PSA(E_j))^n < \prod_{i=1}^{n} PSA(E_i), \ j \in \{1...n\} \tag{10.7}$$

Discussion. Under certain circumstances, a configuration based on homogeneous environments may suffer from attacks with a probability much higher than $PSA_{hom}(E)$. In computer science, monoculture is defined as a community of computers, all running identical software with identical configuration settings, and thus, all of them having the same vulnerabilities and prone to the same attacks. In this sense, a homogeneous environment based on connected computers represents this sort of community. For instance, a home PC connected to the corporative network where a second environment is placed and both of them having the same software and configuration is an example of monoculture scenario.

As mentioned by Geer et al. in [82], a monoculture of networked computers is a convenient and susceptible reservoir of platforms from which to launch attacks, and where these attacks can and do cascade. That is, once one of these computers is infected, the rest can be reached and infected with a probability equal to 1. Thus, the $PSA_{hom}(E)$ is drastically increased to the PSA of just one environment.

According to [82], risk diversification is a primary defense against aggregated risk when that risk cannot otherwise be addressed. In this sense, artificial diversity, a term which covers any technique that creates diversity in information systems for security purposes, was firstly proposed by Forrest et al. in [79] to prevent specific kinds of attacks. Whereas artificial diversity focuses on techniques that modify the same piece of software, true diversity represents platforms where different software runs. Both in artificial and true diversity the result is that the attacker is forced to individualize exploits.

Contrary to this, in [210] the authors present some disadvantages against diversity, defending monoculture as a more robust approach against current threats. In their opinion, deploying diverse systems entails more complex and error-prone configurations, exposing vulnerabilities. Besides, while they accept that monoculture facilitates the spread of attacks in networked computers, they consider that diversification is not always as effective as intended and may involve more complicated testing and debugging procedures, especially in case of artificial diversity.

As extensively demonstrated [82], the risk of monoculture is clear, particularly in case of indiscriminate attacks, that is, attacks that intend to spread as fast and extensively as possible, like Internet malware. In this sense, the analysis given in this Section was developed assuming the existence of independent events, and thus fits with a scenario of indiscriminate attacks. When using diverse systems, the propagation is stopped as soon as the attacker finds a platform which implementation or configuration differs from the one considered in the exploit designed.

The formal proofs of this Section for homogeneous configurations stand as long as the malware cannot automatically reach one environment from another. Otherwise, and as explained above, the division principle is undermined. This requirement does not apply to heterogeneous configurations.

In addition, we find an added benefit in heterogeneous configurations when observing real world scenarios. In practice, users generally use highly risky environments (high PSA), e.g. PC, for their daily operations, like home banking or e-commerce transactions. As a result, claims 2 and 3 are clearly difficult to be achieved. In other words, it is improbable that a homogeneous configuration of such type of environment poses a PSA lower than that of a heterogeneous configuration. As a result, if the user employs a risky environment as one of the environments, using additional heterogeneous environments (e.g. a mobile device) will surely provide a higher level of security.

10.2.1.3 Numerical examples

Tables 10.1 and 10.2 show some numerical examples of the reduction of the resultant PSA when applying the paradigm of the division of the signature environment. Table 10.1 represents some cases of using different number of environments in heterogeneous configurations, while Table 10.2 is focused on homogeneous configurations.

For illustration purposes we consider the next environments, with their corresponding PSA between parenthesis (range between 0 - zero probability of being compromised - and 1 - certainty of having been compromised): E1 (0,1), E2 (0,3), E3 (0,5) and E4 (0,8). We calculate the resultant PSA when applying the division paradigm with a set of 2, 3 and 4 environments, and compare the results with the PSA of a configuration based on a single environment with the lowest PSA, that is, E1.

E1	**E2**	**E3**	**E4**	**Final PSA**	**Reduction/Increase (%)**
0,1	-	-	-	0,1	0 (0%)
0,1	0,3	-	-	0,03	-0,07 (-70%)
0,1	-	0,5	-	0,05	-0,05 (-50%)
0,1	-	-	0,8	0,08	-0,02 (-20%)
-	0,3	0,5	-	0,15	+0,05 (+50%)
-	-	0,5	0,8	0,4	+0,3 (+300%)
0,1	0,3	0,5	-	0,015	-0,085 (-85%)
0,1	0,3	-	0,8	0,024	-0,076 (-76%)
0,1	-	0,5	0,8	0,04	-0,06 (-60%)
-	0,3	0,5	0,8	0,12	+0,02 (+20%)
0,1	0,3	0,5	0,8	0,012	-0,088 (-88%)

Table 10.1: Numerical examples of the PSA reduction when applying the paradigm of the division of signature environment in a heterogeneous configuration

As can be seen in Table 10.1, heterogeneous configurations that add new environments to E1 significantly decrease the resultant PSA (from 20% to 88% of reduction), obtaining more marked reduction as the number of environments increases (from 20% to 70% of reduction for 2 environments, from 60% to 85% of reduction for 3 environments, and 88% of reduction for the 4 environments). It should be noted that some heterogeneous configurations do not decrease the resultant PSA but increase it respecting environment E1 if those environments have a high PSA. The cause is that the reference environment is the one with lowest PSA (E1), which is indeed significantly lower than the PSA of the rest of environments.

E	**E**	**E**	**E**	**Final PSA**	**Reduction/Increase (%)**
0,1	-	-	-	0,1	0 (0%)
0,1	0,1	-	-	0,01	-0,09 (-90%)
0,1	0,1	0,1	-	0,001	-0,099 (-99%)
0,1	0,1	0,1	0,1	0,0001	-0,0999 (-99,9%)
0,3	0,3	-	-	0,09	-0,01 (-10%)
0,3	0,3	0,3	-	0,027	-0,073 (-73%)
0,3	0,3	0,3	0,3	0,0081	-0,0919 (-91,9%)
0,5	0,5	-	-	0,25	+0,15 (+50%)
0,5	0,5	0,5	-	0,125	+0,025 (+25%)
0,5	0,5	0,5	0,5	0,0625	-0,0375 (-37,5%)
0,8	0,8	-	-	0,64	+0,54 (+540%)
0,8	0,8	0,8	-	0,52	+0,42 (+420%)
0,8	0,8	0,8	0,8	0,416	+0,316 (+316%)

Table 10.2: Numerical examples of the PSA reduction when applying the paradigm of the division of signature environment in a homogeneous configuration

On the other hand, Table 10.2 corroborates that increasing the number of environments significantly reduces the resultant PSA, specially when the homogeneous configuration is based on the environment with the lowest PSA, as formally demonstrated in Section 10.2.1.2 (from 90% to 99,9% of reduction).

It is interesting to note how the homogeneous configurations of environment E2 (PSA 0,3), which has a PSA three times as high as the reference environment E1, provide a PSA significantly lower than using the single environment E1 (from 10% to 91,9% of reduction). It is also clear how a homogeneous configuration based on a high PSA (E3 and, particularly, E4) does not reduce, but increases, the resultant PSA respecting E1 (an increase of up to 540%).

10.2.1.4 Towards perfect security

Evidence that cannot be repudiated should provide an irrefutable proof whether the particular event actually took place or not (see Chapters 2 and 3).

Shannon introduced the concept of **perfect security** in [213]. He established that an encryption system is perfectly secure if the attacker does not learn anything at all from observing the ciphertext. In our particular context, we consider that:

A system is perfectly secure if the attacker is completely unable to generate fraudulent evidence on behalf of the user or make fraudulent evidence be verified as valid evidence.

That is, the probability of performing an attack on the system for fraudulent evidence establishment is zero.

The theoretical approach in an environment division principle is as follows. Assuming that the user has access to an infinite number of environments, the PSA in the resultant scenario tends to zero providing that new environments added have a PSA lower than 1:

$$\lim_{n\to\infty} \prod_{i=1}^{n} PSA\,(E_i) \;=\; 0,\; PSA\,(E_i) \;\neq\; 1 \tag{10.8}$$

Though it is a purely theoretical result which put in practice is obviously infeasible, it supports previous claim 1 and demonstrates that the reliability to achieve can be substantially enhanced by adding new environments on demand. On the contrary, there must be a trade-off between the number of environments needed by the user to produce non-repudiation evidence and the pursued reliability. Under the division principle, lower PSA, and thus higher reliability of evidence, implies a higher number of environments. The decision regarding the number of environments needed should be made on a case-by-case basis.

10.2.2 Analysis respecting the taxonomy

In this Section we analyze and calculate the theoretical probability of a successful attack on evidence establishment and for each attack category identified in the taxonomy presented in Chapter 6. The analysis is performed considering two different circumstances: if our thesis proposal is applied, and if it is not.

In the analysis, we also provide a reasoning of the capability of our proposal to counteract the attack categories of the taxonomy. In particular, our analysis is focused on the methods of attacks that can be used by the attacker to subvert the reliability of a digital signature generation or verification process. Therefore, we will take into account only the dimension two of the taxonomy, not including the dimension one (goal) and dimension three (target). For usefulness purposes, only subcategories that describe the specificity of the attack method are considered, being marked in bold. For example, analyzing category *D2-CAT2: Modification prior to signature computation* is of no use as more specific subcategories exist, and that provide more information for the analysis.

For each subcategory, the probability of a successful attack (PSA) of such method on a signature creation environment (SCE) or signature verification environment (SVE) that implements our proposal (notation $Set(E)[SCE]$ or $Set(E)[SVE]$) is calculated, and compared with the PSA on a single environment (notation $E[SCE]$ or $E[SVE]$). The PSA is calculated for SCE, SVE or both depending on the method of attack (see Table 6.2, which establishes the relationship between the dimension *Attacker's Goal* and dimension *Method of Attack*).

It should be noted that the PSA calculation and comparison is from a theoretical viewpoint, not using concrete values. Some subcategories are grouped for the analysis if the reasoning and PSA calculation is the same.

Our proposal consists of the next thesis contributions, indicating an acronym for each one for simplicity purposes during the evaluation:

- Division of the Signature Environment, DSE (Chapter 7)
- Conscious data verification, CDV (Chapter 7)
- Restricted format of data to be signed, RFDTBS (Chapter 7)
- Extended electronic signature policies, EXTSP (Chapter 8)
- Certificate extension, CE (Chapter 8)

D2-CAT1: Environment manipulation

The formal PSA reduction is achieved due to DSE, CDV and EXTSP. CDV helps detecting, at any of the environments, any undesired change during the evidence generation or verification.

$$\boxed{PSA\,(Set\,(E[SCE])) \;=\; PSA\,(Set\,(E[SVE])) \;=\; \prod_{i=1}^{n} \; PSA\,(E_i)}$$

On the contrary, the formal PSA for a single environment is as follows.

$$\boxed{PSA\,(E[SCE]) \;=\; PSA\,(E[SVE]) \;=\; PSA\,(E)}$$

In this case the enhancement is theoretically demonstrated by simply comparing the obtained PSAs. The practical enhancement will depend on the number and nature of the environments used.

D2-CAT2: Modification prior to signature computation

D2-CAT2.1: Document modification

D2-CAT2.1.1: Dynamic content inclusion, and all subcategories

D2-CAT2.2: Attribute modification

D2-CAT2.2.1: Dynamic content inclusion, and all subcategories

The DSE, CDV and EXTP are not effective against this type of attack, as the attacker only needs to compromise the first environment in order to include the dynamic content into the data to be signed (DTBS), either the document or attributes. The modified DTBS would then be transmitted from one environment to another, even though these environments had not been compromised by the attacker. If the dynamic content could be enforced by the attacker at will and once the evidence had been completed, the user would not notice that an attack had been performed.

Therefore, in this case the enhancement is achieved thanks to RFDTBS, which prevents the attacker from including dynamic content into the DTBS.

$$\boxed{PSA\,(Set\,(E[SCE])) \;=\; 0}$$

On the other hand, the formal PSA for a single environment remains as follows.

$$\boxed{PSA\,(E[SCE]) \;=\; PSA\,(E)}$$

It should be noted that PSA above would be 0 as well if RFDTBS was applied as a requirement for the single SCE.

D2-CAT2: Modification prior to signature computation
D2-CAT2.1: Document modification
D2-CAT2.1.2: Content modification
D2-CAT2.2: Attribute modification
D2-CAT2.2.2: Content modification
D2-CAT2.3: DTBS modification
D2-CAT2.4: DTBSR modification

The formal PSA reduction is achieved due to DSE, CDV and EXTSP. CDV helps detecting, at any of the environments, any undesired change during the evidence generation.

$$\boxed{PSA\,(Set\,(E[SCE])) \;=\; \textstyle\prod_{i=1}^{n}\; PSA\,(E_i)}$$

On the contrary, the formal PSA for a single environment is as follows.

$$\boxed{PSA\,(E[SCE]) \;=\; PSA\,(E)}$$

In this case the enhancement is theoretically demonstrated by simply comparing the obtained PSAs. The practical enhancement will depend on the number and nature of the SCEs used.

D2-CAT3: Modification post signature computation
D2-CAT3.1: External content

The DSE, CDV and EXTSP are not effective against this type of attack, as the attacker only needs to modify the referenced content once the evidence has been generated. As a result, only RFDTBS can counteract this method of attack. RFDTBS prevents the user from including a reference to external content in the data to be signed, being this attack completely thwarted.

$$\boxed{PSA\,(Set\,(E[SCE])) \;=\; 0}$$

On the other hand, the formal PSA for a single environment remains as follows.

$$\boxed{PSA\,(E[SCE]) \;=\; PSA\,(E)}$$

Again, it should be noted that PSA above would be 0 as well if RFDTBS was applied as a requirement for the single SCE.

D2-CAT3: Modification post signature computation
D2-CAT3.2: Cryptanalysis, and all subcategories

An attack focused on finding a collision in the hash function used for the digital signature computation highly depends on the internal structure of the hash function. Design-independent attacks basically consist of brute force attacks, what may lead to a complexity of $O(2^{n/2})$ (worst case) if the hash function is collision resistant (we assume that a collision resistant hash function is used for both a set of environments and a single environment).

Therefore, and assuming $P(H_i)$ as the probability to find a collision in hash function H_i, we can establish the next PSA as the resultant PSA in case a different hash function is used in each environment of $Set(E)$.

$$\boxed{PSA\,(Set\,(E[SCE])) \;=\; \prod_{i=1}^{n} \; P\,(H_i)}$$

In case of reusing the same hash function in different environments, the resultant PSA increases accordingly.

On the other hand, the formal PSA for a single environment is the next.

$$\boxed{PSA\,(E[SCE]) \;=\; P\,(H)}$$

As can be seen, the DSE does not enhance the reliability of evidence on its own, but in conjunction with the diversity concept applied to the hash functions used. This requirement can be established at signature policy level, specifying the algorithms acceptable for each digital signature.

D2-CAT4: Unauthorized invocation of the signing function
D2-CAT4.1: Compromise of the signer authentication data (SAD)
D2-CAT4.1.1: Social engineering

Although the PSA reduction might seem difficult to be formalized at first glance, as the success of social engineering techniques highly depends upon personal and social characteristics of the users, it is actually achieved due to DSE, CDV, EXTSP and CE. Depending on the number of (S)SCDev being used by the signer, and that in turn depends on the way that the environment and the signature creation data (SCD) have been bound to each other (see Chapter 7), the reduction is justified using a different rationale.

(a) If the SCD and the environment are bound to each other by means of the environment attestation technique, then the signer could use a single (S)SCDev to compose the valid evidence. Though the attacker might only need to obtain one SAD, it would still need to compromise each environment to generate the valid evidence.

(b) On the contrary, if the binding is established by procedure, then the user is forbidden to share the same SCD between different environments. Consequently, and as well as the attacker would need to obtain as many SADs as established by the procedure (what would possibly make the signer suspect that an attack is being carried out), it would need to compromise every environment to generate the valid evidence.

Finally, CE contributes by preventing an attacker that has compromised a single environment to generate binding signatures on behalf of the signer, while CDV permits the signer to detect any evidence generation started from a compromised environment. Both in (a) and (b), the formal PSA reduction remains as follows:

$$\boxed{PSA\,(Set\,(E[SCE])) \;=\; \prod_{i=1}^{n}\; PSA\,(E_i)}$$

On the other hand, the formal PSA for a single environment is as follows. It should be mentioned that, in this case, the PSA highly depends on the capability to obtain the SAD from the signer, which in turn depends upon personal and social characteristic of the signer.

$$\boxed{PSA\,(E[SCE]) \;=\; PSA\,(E)}$$

D2-CAT4: Unauthorized invocation of the signing function
D2-CAT4.1: Compromise of the signer authentication data (SAD)
D2-CAT4.1.2: SAD interception, and all subcategories
D2-CAT4.1.3: Guessing
D2-CAT4.2: Authentication Bypass
D2-CAT5: Compromise of the signature creation data (SCD), and all subcategories, except D2-CAT5.4: Cryptanalysis

The formal PSA reduction is achieved due to DSE, CDV, EXTSP and CE. CDV helps detecting, at any of the environments, any undesired evidence generation started by the attacker. CE contributes by preventing an attacker that has compromised a single environment or SCD from generating binding signatures on behalf of the signer.

$$\boxed{PSA\,(Set\,(E[SCE])) \;=\; \prod_{i=1}^{n}\; PSA\,(E_i)}$$

On the contrary, the formal PSA for a single environment is as follows.

$$\boxed{PSA\,(E[SCE]) \;=\; PSA\,(E)}$$

In this case the enhancement is theoretically demonstrated by simply comparing the obtained PSAs. The practical enhancement will depend on the number and nature of the SCEs used.

D2-CAT5.4: Cryptanalysis

When this method of attack is applied, the PSA reduction is derived in the same way as in *D2-CAT3.2: Cryptanalysis*. Therefore, and assuming $P(PKC_i)$ as the probability to obtain the private key (SCD) used in certain public key algorithm PKC_i, we can establish the next PSA as the resultant PSA in case a different SCD is used in each environment of $Set(E)$.

$$\boxed{PSA\,(Set\,(E[SCE])) \;=\; \prod_{i=1}^{n}\; P\,(PKC_i)}$$

In this case, and contrary to *D2-CAT3.2: Cryptanalysis*, the resultant PSA remains the same due to the binding principle between the SCD and the environment. That is, if the user reuses the same SCD in every environment, then a binding by environment attestation is mandatory, implying that each environment uses a different SCD. Otherwise, if a binding by procedure is applied, then the user is required to use a different SCD at each environment.

On the other hand, the formal PSA for a single environment is the next.

$$\boxed{PSA\,(E[SCE]) \;=\; P\,(PKC)}$$

Like in hash functions, the DSE enhances the reliability of evidence in conjunction with the diversity concept applied to the SCD used, implemented in this case by means of the binding principle.

D2-CAT6: Influence on certificate verification result

D2-CAT6.1: Alteration of subscriber's revocation request, and all subcategories

Though these methods of attack aim at subverting the verification process, the attacks themselves must be carried out on the SCE. The formal PSA reduction is achieved due to DSE, with a particular rationale given further.

$$\boxed{PSA\,(Set\,(E[SCE])) \;=\; m/n \;*\; (\prod_{i=1}^{m}\; PSA\,(E_i)),\text{ with } m \geq 1 \text{ and } n \geq 2}$$

The formula above is a generalization by which the attacker chooses m environments selected among all environments (n), with a probability m/n of having chosen the right environment from which the revocation will be requested. Obviously, if our proposal is applied, then n must be greater than or equal to 2. In addition, it is required that the attacker compromises each selected environment.

On the other hand, the formal PSA for a single environment is represented next, and coincides with the case of m and n having the value of 1. It is being assumed that the user owns a single environment from which generating the signature and requesting the certificate revocation.

$$\boxed{PSA\,(E[SCE]) \;=\; PSA\,(E)}$$

Independently of which is the value of m in the formula above, the resultant $PSA\,(Set\,(E[SCE]))$ will always be lower than $PSA\,(E[SCE])$.

D2-CAT6: Influence on certificate verification result
D2-CAT6.2: Alteration of certificate status verification
D2-CAT6.2.1: Grace or cautionary period bypassing, and all subcategories

Again, though these methods of attack are focused on subverting the verification process, they imply that the signature creation data (SCD) needed to generate the evidence has been previously compromised. Even subcategory *D2-CAT6.2.1.3: Exploit delay in CA's revocation request processing* assumes that the signer can have noticed the SCD compromise and requested the certificate revocation. Therefore, the formal PSA reduction is achieved due to DSE and EXTSP.

$$\boxed{PSA\,(Set\,(E[SCE])) \;=\; \prod_{i=1}^{n}\; PSA\,(E_i)}$$

The formal PSA for a single environment is as follows.

$$\boxed{PSA\,(E[SCE]) \;=\; PSA\,(E)}$$

In this case the enhancement is theoretically demonstrated by simply comparing the obtained PSAs. The practical enhancement will depend on the number and nature of the SCEs used.

D2-CAT6: Influence on certificate verification result
D2-CAT6.2: Alteration of certificate status verification, and all subcategories, except D2-CAT6.2.1: Grace or cautionary period bypassing
D2-CAT6.3: Untrusted trust anchor/trust point addition
D2-CAT6.4: Alteration of certificate integrity verification result
D2-CAT6.5: Alteration of certificate validity period verification result
D2-CAT7: Influence on signature verification result, and all subcategories

The formal PSA reduction is achieved due to DSE and EXTSP. It is assumed that any modification of any message is carried out within the boundaries of the SVE.

$$\boxed{PSA\,(Set\,(E[SVE])) \;=\; \prod_{i=1}^{n} PSA\,(E_i)}$$

The formal PSA for a single environment is as follows.

$$\boxed{PSA\,(E[SVE]) \;=\; PSA\,(E)}$$

In this case the enhancement is theoretically demonstrated by simply comparing the obtained PSAs. The practical enhancement will depend on the number and nature of the SVEs used.

10.3 Experimental Implementation

This Section explains the experimental implementation developed, which includes the validation algorithm of the extended electronic signature policies framework, and a simulator for OFEPSP+, where the validation algorithm is used and a virtual division of the origin's environment applied. An overview is given first in Section 10.3.1, covering the software architecture, the electronic signature formats used and the set of signatures composition. The results obtained from the evaluation of the validation algorithm are provided in Section 10.3.2. Finally, Section 10.3.3 contains the description of the simulator for OFEPSP+.

10.3.1 Overview

In order to evaluate the correctness and feasibility of the validation algorithm and OFEPSP+ protocol, an experimental implementation has been developed.

In particular, the implementation covers the validation process of a set of signatures in accordance with the requirements established in an extended electronic signature policy. This procedure is specified in Section 8.2.2, while the pseudo-code of the validation algorithm developed is detailed in Appendix C. The procedure explained in Section 8.2.1, where the generation of a signature according to the extended policy is explained, has not been developed. Additionally, a simulator for OFEPSP+ has been developed. The simulator instantiates each participant of the protocol, including two virtual environments for the origin, and supports the main protocol and the abort and recovery subprotocols.

The implementation has been made in Java language. Afterwards, the validation algorithm has been tested using the JUnit Test Framework [128], while OFEPSP+ simulator has been tested by running different scenarios, including a normal protocol execution (no participation of TTP is needed), and two different executions where the abort and recovery subprotocols have to be executed.

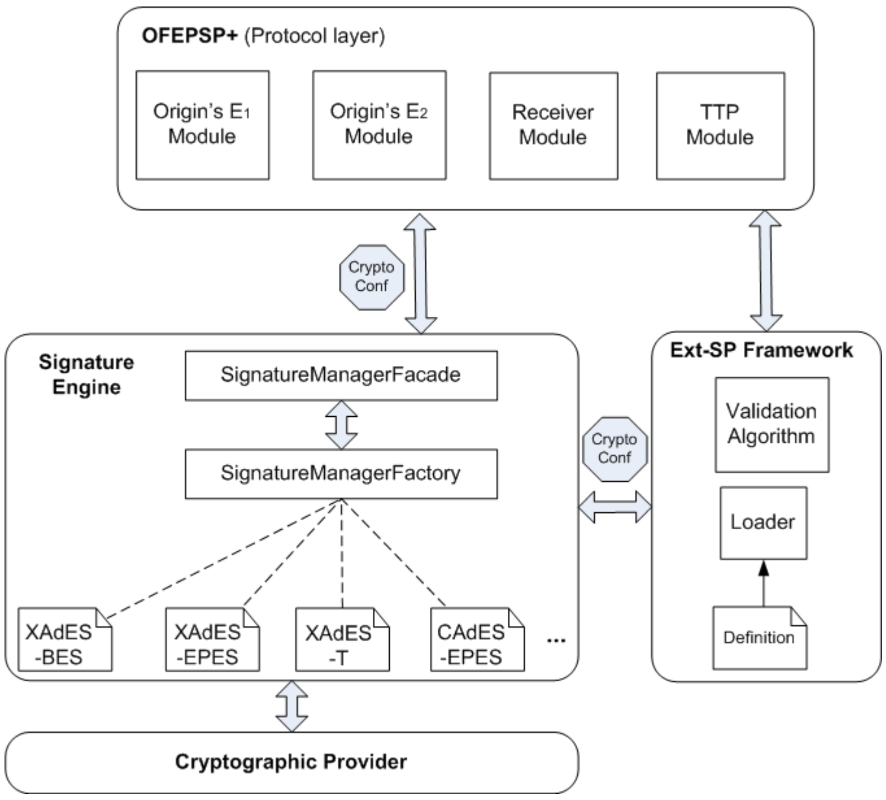

Figure 10.7: Overview of the architecture.

10.3.1.1 Architecture

Figure 10.7 shows the software layer architecture of the experimental implementation.

The Extended Electronic Signature Policies (ext-SP) framework provides a generic and protocol-independent service to higher layers. For that reason, the framework has been implemented and separately tested from OFEPSP+. As can be seen in the Figure, the framework is limited to the validation algorithm and an auxiliary module that loads the definition of the tree of signatures (TSi) to be used by the algorithm.

The signature engine implements the electronic signature generation and validation capabilities. Although the current experimental implementation deals only with XAdES-EPES signatures, the engine has been designed to permit the integration of other formats (e.g. XAdES-T, CAdES-BES, CAdES-EPES, etc.) in a transparent way.

Both the ext-SP framework and the protocol layer (OFEPSP+ in particular) access the signature capabilities through a generic facade which is independent from the particular signature format instantiated. The features needed by the factory to instantiate a signature format implementation are indicated in the cryptographic configuration established by the layer invoking the signature engine. The cryptographic configuration includes features like the particular signature format to be instantiated, the cryptographic algorithms to be used by the signature implementation or the signature policy information that orchestrates the generation and validation of the signature (in case of AdES-EPES formats).

Finally, IAIK [109] has been used as the Cryptographic Provider. IAIK implements the low level cryptographic capabilities according to the Java Cryptographic Architecture and Java Cryptographic Extension [126]. Two IAIK toolkits have been used:

- IAIK *XML Security Toolkit (XSECT)* [105], which implements the upcoming APIs for the Java platform: XML Digital Signatures APIs as specified by the Java Specification Request JSR#105 ; and XML Digital Encryption APIs (not used in the experimental implementation) as specified by JSR#106.
- IAIK *XML Advanced Electronic Signatures (XAdES) add-on* for XSECT [104], which enables the creation of XAdES signatures according to ETSI TS 101 903 V1.3.2 standard [75].

10.3.1.2 Electronic signature format

As mentioned above, the design of the experimental implementation supports the addition of any electronic signature format in a transparent manner. However, and because the goal of the experimental implementation is to serve as the basis for the validation algorithm and protocol evaluation, we have restricted the supported formats to the XML Advanced Electronic Signature (XAdES) format, as specified in [75]. More specifically, we have chosen the Explicit Policy based Electronic Signature (XAdES-EPES) format.

The XAdES-EPES extends the definition of a basic electronic signature (XMLDSig or XAdES-BES) to conform to the identified signature policy. Therefore, this format is linked to the usage of a signature policy. However, the standard allows the inclusion of optional information when creating the signature. Next, the information included in the XAdES-EPES signatures used in our experimental implementation is given. Details about the XAdES-EPES format can be found in the standard [75].

Because the signature format supported by the experimental implementation is based on XML, then the extended signature policies had to be defined in XML as well. Therefore, policies defined in ASN.1 have not been implemented. Please refer to Appendix E for the detailed XML Schema Definition (XSD) of the policy.

Signing time

If a dependence, either absolute or relative, exists between two or more signatures (indicated in the ext-SP), then a time reference must be used to determine whether the dependence is fulfilled or not. The most reliable way of doing so is to use timestamps issued by a trusted Time-Stamping Authority (TSA) [5], and included in the signature as the *xades:SignatureTimeStamp* unsigned property. If a TSA is not available, the time reference can be taken from the optional signed property named signing time (*xades:SigningTime*), which specifies the time at which the signer (purportedly) performed the signing process. Generally, the clock of the computer is used as the source of time.

The source of time to use is not a decision that can be made by the signer nor the verifier. It is the signature policy the place where this requirements is established. Therefore, if the policy obliges to use a trusted TSA, and the signer or the verifier does not include a timestamp in the signature, then the signature verification must fail. However, and for the experimental implementation, we have discarded the usage of a TSA, using the clock of the computer for producing the signing time value, as defined in Section 7.2.1 of the XAdES standard [75]. The reason lies in that most test cases demand an accurate and precise set of signatures in terms of the time at which each signature is generated. For instance, some TSis used in certain test cases oblige a signer to generate the signature minutes or even hours after another one was computed. Therefore, the generation of the corresponding SSis (both for success and error cases) would have implied an unfeasible delay during the composition of most set of signatures for the test bench. As a result, we consider that, for the evaluation of the experimental implementation, XAdES-EPES signatures including the *xades:SigningTime* signed property suffices.

Signing certificate

XAdES standard establishes in Section 4.4.1 that it is mandatory to protect the signing certificate with the signature, in one of two ways: either incorporating the signed property called *xades:SigningCertificate* or not incorporating it but including the signing certificate within the *ds:KeyInfo→ds:X509Data→ds:X509Certificate* element and signing it. In our experimental implementation we have developed both alternatives. Our need was actually being able to access the whole certificate in order to retrieve the subject distinguished name information, which is used by the extended signature policy framework for establishing the mappings between the *subjectDNs* and the nodes' signer's identifiers.

Signature policy

We include the mandatory *xades:SignaturePolicyIdentifier* signed property, which definition is given in Section 7.2.3 of the XAdES standard. This element explicitly includes the signature policy identifier information. The explicit identifier form consists of the policy identifier and the hash of the document where the policy is specified. We have not added any optional policy qualifier.

For test purposes, we have used a fixed value for the signature policy identifier: `http://jlopez.thesis.uc3m.es/SigPolicy/vXXX`. In order to make different signers follow different policies, we just modify the version (vXXX) of the policy in the URI (e.g. v1.0, v1.0.1, etc.).

Extended signature policy

Chapter 8 proposes the new signed property named *ExtSignaturePolicyIdentifier* as a reference to the extended signature policy to which the signer adheres when creating the signature. However, and in the experimental implementation, we do not generate XAdES-EPES signatures containing that new property. The reason is that IAIK, the underlying cryptographic provider, does not obviously allow the addition of properties not compliant to the standard when creating the signatures. Furthermore, and contrary to other data (e.g. the signature policy reference, the commitment type, the signing time, etc.), the ext-SP reference is a fixed value that we do not need to modify in our test cases. Therefore, this limitation does not reduce the accuracy of the tests results. Notwithstanding, it would be necessary to achieve a solution if a prototype of the framework had to be developed in a future.

Commitment type

We include the optional *xades:CommitmentTypeIndication* signed property, as defined in Section 7.2.6 of the XAdES standard. The type of commitment made by the signer when producing the signature is indicated in an explicit manner by means of an URI value. A number of commitments have already been identified in the standard [75]:

- **Proof of origin** indicates that the signer recognizes to have created, approved and sent the signed data object. The URI for this commitment is `http://uri.etsi.org/01903/v1.2.2#ProofOfOrigin`.
- **Proof of receipt** indicates that signer recognizes to have received the content of the signed data object. The URI for this commitment is `http://uri.etsi.org/01903/v1.2.2#ProofOfReceipt`.
- **Proof of delivery** indicates that the Trusted Service Provider (TSP) providing that indication has delivered a signed data object in a local store accessible to the recipient of the signed data object. The URI for this commitment is `http://uri.etsi.org/01903/v1.2.2#ProofOfDelivery`.

- **Proof of sender** indicates that the entity providing that indication has sent the signed data object (but not necessarily created it). The URI for this commitment is `http://uri.etsi.org/01903/v1.2.2#ProofOfSender`.
- **Proof of approval** indicates that the signer has approved the content of the signed data object. The URI for this commitment is `http://uri.etsi.org/01903/v1.2.2#ProofOfApproval`.
- **Proof of creation** indicates that the signer has created the signed data object (but not necessarily approved, nor sent it). The URI for this commitment is `http://uri.etsi.org/01903/v1.2.2#ProofOfCreation`.

Additionally, and for test purposes only, we have defined the next complementary commitment types, with their corresponding (fictitious) URIs:

- **Proof of storage**, with the next value as the URI `http://uri.etsi.org/01903/v1.2.2#ProofOfStorage`
- **Proof of acknowledgment**, with the next value as the URI `http://uri.etsi.org/01903/v1.2.2#ProofOfAcknowledgment`
- **Proof of review**, with the next value as the URI `http://uri.etsi.org/01903/v1.2.2#ProofOfReview`
- **Proof of second review**, with the next value as the URI `http://uri.etsi.org/01903/v1.2.2#ProofOfSecondReview`
- **Proof of third review**, with the next value as the URI `http://uri.etsi.org/01903/v1.2.2#ProofOfThirdReview`
- **Proof of fourth review**, with the next value as the URI `http://uri.etsi.org/01903/v1.2.2#ProofOfFourthReview`
- **First partial non-repudiation of origin**, with the next value as the URI `http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfOrigin1`
- **First partial non-repudiation of receipt**, with the next value as the URI `http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfReceipt1`
- **Second partial non-repudiation of origin**, with the next value as the URI `http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfOrigin2`
- **Second partial non-repudiation of receipt**, with the next value as the URI `http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfReceipt2`

- **Non-repudiation evidence**, with the next value as the URI `http://uri.etsi.org/01903/v1.2.2#NonRepudiationEvidence`

10.3.1.3 Set of signatures composition

According to Section 7.2.6, we restrict the format of the data to be signed to XML (though it may contain external references such as DTD, XML schema, etc.), XML elements mixed with ASCII text, ASCII text content only and binary information. In this last case, the information is encoded in Base64 format in order to avoid problems with the parsing layers and intermediate entities, when transmitted through the network.

The Primary Signatures are collected in a detached form, so they are detached from the data that is signed. As the XML Signature standard [233] does not impose any requirement on how detached XML signatures have to be linked to the signed data (e.g. collected in the same XML document along with the signed data, or in a separate document), we compose an XML document with a root node named *Root* and a child node named *SignedData*. The data to be signed is inserted in the document as a child node of the *SignedData* node. Besides, each Primary Signature is included in the document as a direct child of the root node. The CounterSignatures are included as enveloped signatures of the corresponding countersigned signature. According to the standard [75], the CounterSignature can be incorporated as an unsigned signature property of the countersigned signature.

Thereby, we compose the set of signatures (SSi) that is used by the validation algorithm as the XML document, containing both the signed data and the XAdES-EPES signatures. Next, the XML schema of the SSi herein considered is indicated:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <xsd:element name="Root" type="ssiType"/>
  <xsd:complexType name="ssiType">
    <xsd:sequence>
      <xsd:element name="SignedData" type="AnyType"/>
      <xsd:sequence>
        <xsd:element name="Signature" type="ds:SignatureType"
                      maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="AnyType" mixed="true">
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
```

```
      <xsd:any namespace="##any" processContents="lax"/>
    </xsd:sequence>
    <xsd:anyAttribute namespace="##any"/>
  </xsd:complexType>
</xsd:schema>
```

As can be seen in the schema, we use the *AnyType* schema data type, which is defined in [75]. It has a content model that allows a sequence of arbitrary XML elements that (mixed with text) is of unrestricted length. It also allows for text content only. Additionally, an element of this data type can bear an unrestricted number of arbitrary attributes.

It is worth to mention that other variants of composition, compliant with the standard as well, could be used:

- *Enveloping signatures*, where the Primary Signature is generated over the content found within an Object element of the signature itself. Thus, the signed data is a child element of the signature. In our particular case, and because several signatures are supposed to be generated, using enveloping signatures would imply the resultant SSi to be of significant greater size, as the content is replicated as an embedded Object for each Primary Signature.

- *Enveloped signatures*, where the signature is generated over the XML content that contains the signature as a child element. If multiple signature are to be generated, then each signature must omit itself from its own calculations, but it is also necessary to exclude the ongoing signatures from the calculations of the previously generated signatures so that adding the current signature does not break the previous signatures' integrity.

- *Detached countersignatures*, where the CounterSignature is detached from the signature it countersigns, instead of being a child element included as an unsigned property.

- Combinations of the above.

Our decision for the composition of the set of signatures using detached Primary Signatures and enveloped CounterSignatures lies in the fact that it is optimum respecting the resultant length, the calculation of signatures is much easier than if complex XPaths formulas have to be used (i.e. in enveloped signatures), and the whole information is collected in an self-contained document (the SSi), easing verification tasks.

10.3.2 Evaluation of the validation algorithm

This Section demonstrates the correctness of the validation algorithm proposed in the framework for extended signature policies by summarizing the test cases executed and the results obtained. This algorithm is the most complex and sensitive part of the framework (see Section 8.2.2 and Appendix C, where the validation algorithm is explained in detail). The procedure explained in Section 8.2.1, where the generation of a signature according to the extended policy is explained, has not been developed.

The validation of a signature against the requirements imposed in a simple signature policy has not been developed either. From a research viewpoint, the cost of developing such functionality is not worth the added value provided to the experimental implementation nor to the evaluation of the proposal contained in this thesis. As could be seen in Chapter 8, the validation algorithm should validate each signature against its signature policy either at the very beginning or as the very last step. Therefore, at those points, the algorithm either has not started or has just finished the necessary operations to discover if the tree represented by the Set of Signatures (SSi) satisfies at least one tree represented by a Tree of Signatures (TSi) according to the extended policy. The unitary signature validation respecting its signature policy has to be obviously carried out before ascertain the complete compliance, but it can be considered out of the scope of the framework goal. As a result, in the experimental implementation we suppose that each signature is compliant with the referenced signature policy.

The defined test cases check that the functional requirements of the validation algorithm are met. In particular, each test case checks that the validation of a SSi against a TSi defined under an extended electronic signature policy matches the expected result.

Several test cases have been defined in order to obtain a high assurance respecting the correctness of the design and implementation of the algorithm. The total number of test cases executed is 188, **all of them fulfilling the expected results**. Each test case has been designed to meet a particular objective. The objective is mainly the functionality that is desired to be tested (e.g. a pruning, correct backward processing, correct dependences evaluation, deadlocks in different circumstances, etc.), besides obviously checking that the specific case set out was correctly evaluated by the algorithm. JUnit has been used as the Test Framework, and each test case executes the validation algorithm with a SSi and a TSi as parameters.

The test bench used for the tests cases, and that consists of a set of TSis and a set of SSis, is given in Appendix D. According to the specification given in Chapter 8, the information used for each TSi node includes the node's unique identifier, the node's signer identifier, the allowed commitment types, the acceptable signature policies, the countersignatures (child nodes) and the timing and sequence dependences. The SSi consists of an XML document with the signed data and a collection of one or more XAdES-EPES signatures, as explained in Section 10.3.1.3.

The types of TSis and SSis created and used as the test bench are summarized next. Details of each TSi and SSi composition are given in Appendix D:

- TSis with one Primary Signature (see Table D.2).
- TSis with three Primary Signatures (see Table D.3).
- TSis with one Primary Signature and one CounterSignature each (see Table D.4).
- TSis with several levels of depth (see Table D.5).
- TSis specifically designed for OFEPSP+ protocol (see Table D.6).
- SSis with one Primary Signature (see Table D.7).
- SSis with three Primary Signatures (see Table D.8).
- SSis with several levels of depth (see Table D.9).
- SSis specifically designed for OFEPSP+ protocol (see Table D.10).

From the information above, test cases have been grouped according to the type of SSi being analyzed. This SSi is further evaluated against a number of TSis:

- Test cases defined for SSis with one level of depth and one Primary Signatures (see Table D.11).
- Test cases defined for SSis with one level of depth and three Primary Signatures (see Table D.12).
- Test cases defined for SSis with several levels of depth (see Table D.13).
- Test cases specifically defined for OFEPSP+ protocol (see Table D.14).

The details of each test case are also given in Appendix D, providing specific information regarding the SSi and TSi used, the result obtained after the execution of the test case, and a brief description of the cause or reason in case the SSi does not satisfy the TSi.

10.3.3 OFEPSP+ simulation

A simulator for OFEPSP+ has been developed in order to verify the executability and feasibility of the protocol, including the main protocol and the abort and recovery subprotocols. The simulator also uses the implementation of the validation algorithm to check the validity of the SSi generated after a protocol run against the requirements imposed in a TSi specifically defined for OFEPSP+.

Four independent Java applications have been implemented, simulating the behavior of each participant in a protocol run. Each participant (origin, receiver, TTP) has been represented by one application, except the origin, that has been split into two applications, one for each of the environments used.

During an OFEPSP+ run, the simulator automatically executes the steps that the origin, receiver and TTP have to follow in order to complete the protocol execution in accordance with its specification and the restrictions and format established in the experimental implementation. Partial evidence is generated in each step, until an extended signature policy compliant SSi is generated. Depending on the step being executed, the signature is generated as a primary signature ($PNRO_1$, $PNRR_1$) or as an embedded countersignature ($PNRO_2$, $PNRR_2$, NRE, NRE_{TTP}).

The Java applications for the receiver and the TTP implement a server socket that listens in a specific IP address and port and accepts requests coming from origins or origins/receiver, respectively. On the other hand, the Java applications for environments E_1 and E_2 of the origin implement a client socket to connect to the receiver for the protocol execution. Additionally, environment E_1 is capable of connecting to the TTP in case the abort subprotocol had to be executed, and configured at application level with a timeout option. The receiver application also implements a client socket to communicate with the TTP in case the recovery subprotocol had to be executed, and configured with a timeout option as well.

Next three scenarios have been simulated:

- An OFEPSP+ execution where origin and receiver communicate with each other in a normal way, without interacting with the TTP.
- An OFEPSP+ execution where we simulate a malicious behavior from the receiver's side. In this case, the receiver does not send $PNRR_2$ to E_1, but executes the recovery subprotocol in order to obtain NRE_{TTP} from the TTP. As the origin does not receive $PNRR_2$, he invokes the abort subprotocol from E_1. Because the receiver executed the recovery subprotocol first, the origin obtains NRE_{TTP} from the TTP.

- An OFEPSP+ execution where we simulate a network communication error that forces the subprotocols to be executed. In particular, the receiver sends $PNRR_2$ to E_1, but we simulate a message loss, forcing the origin to abort the protocol from E_1. Afterwards, and because the receiver does not receive NRE, he executes the recovery subprotocol, obtaining the abort token from the TTP.

10.4 OFEPSP+ Formal Validation

Formal validation of security protocols is of utmost importance before they gain market or academic acceptance. Some standard and widely used security protocols for the Internet have been proved to suffer from critical design flaws that an attacker can exploit to subvert their security. The reason is that their security goals were merely informally evaluated, obviating potential attack paths. Automated reasoning techniques are commonly used to evaluate the protocols in a formal way, increasing the assurance respecting the purported security. In this sense, the Automated Validation of Internet Security Protocols and Applications (AVISPA) [15] and the Security Protocol ANimator for AVISPA (SPAN) [85] tools have been used to validate the correctness and safety of the improved Optimistic Fair Exchange Protocol based on Signature Policies (OFEPSP+) proposed in Chapter 9.

AVISPA [13, 15] provides a suite of applications for building and analyzing formal models of security protocols. AVISPA incorporates four backends: the On-the-Fly Model-Checker (OFMC) [25], the Constraint-Logic-based model-checker (CL-AtSe) [228], the SAT-based Model-Checker (SATMC) [14], and the Tree Automata based Automatic Approximations for the Analysis of Security Protocols (TA4SP) [37]. These modules implement different automated reasoning techniques to formally analyze the protocol specification. On the other hand, SPAN [85] offers a graphical user interface that allows the protocol designer to easily interact with AVISPA capabilities.

Protocol models must be written in the High Level Protocol Specification Language (HLPSL) [16, 49], which allows the protocol designer to describe the security protocol and specify its intended security properties. HLPSL details for OFEPSP+ can be found in Appendix G.

AVISPA adopts the standard intruder model of Dolev and Yao (DY model) [66], in which the intruder has complete control over the network but cannot break cryptography. The intruder may intercept, analyze, and/or modify messages (as far as he knows the required keys), and send any message he composes to whoever he pleases, posing as any other agent. The goal of OFEPSP+ validation was to check the correctness and safety of the protocol respecting DY model.

The validation methodology followed can be summarized in the next 3 steps, which are covered in subsequent Sections:

1. OFEPSP+ specification in HLPSL.
2. HLPSL correctness verification.
3. OFEPSP+ security validation.

Due to some limitations found in the backend analyzers, further explained, we could not formally validate fairness as a security goal of the protocol. For that reason, Section 10.4.4 includes an informal analysis of the security of OFEPSP+ regarding the fairness property.

10.4.1 OFEPSP+ specification in HLPSL

OFEPSP+ complexity lies in the existence of four entities. The first two entities, E_1 and E_2, are managed by the origin of the protocol, but, in HLPSL, had to be considered as two different roles played by a different agent each, named e_1 and e_2 respectively. In any case, the knowledge sharing permitted by HLSP, and the fact that both entities actually correspond to physically separated environments, make the protocol modeling accurate respecting a real scenario. The other two entities, R (the receiver) and the TTP (Trusted Third Party), were modeled as the *receiver* and *server* roles, respectively, played by the corresponding agents.

Each role implemented its own state transition system according to the steps indicated in the protocol described in Sections 9.1.3, 9.1.4 and 9.1.5. Thanks to the *set* structure available in HLPSL, the TTP's evidence database could be modeled and the TTP behavior approached.

It should be remarked that, for simplicity purposes, the HLPSL specification of the protocol assumes the binding by procedure, and not the environment attestation technique applied in Chapter 9.

10.4.1.1 Restrictions applied

The protocol steps described in Sections 9.1.3, 9.1.4 and 9.1.5 could be modeled, except next four issues.

Signature policy-based design

OFEPSP+ fairness property is enforced by the signature policy-based design, assuring that partial evidence does not tie down any of the parties involved. The existence of NRE or NRE_{TTP} is imperative to make the commitment made by both parties binding, and its creation (by the origin or the TTP) must fulfill the policies constraints and requirements. We found that the usage of signature policies could not be modeled in HLPSL. HLPSL allows translating an Alice & Bob chart into a more detailed and expressive language. However, not every protocol behavior can be mapped in HLPSL. Therefore, the protocol steps related to a policy retrieval were discarded during the HLPSL definition.

Time-stamping

Time-stamps can be used in the protocol as a time reference to permit the parties to decide when the recovery or abort subprotocols have to be executed. However, HLPSL only allows the establishment of a generic timeout as an incoming message to a role. This feature avoided us to model specific timeouts, and thus, the time-stamping processes were obviated. We have checked that this constraint has not modified neither the protocol behavior nor its security goals fulfillment.

Template usage

In OFEPSP+ the origin must create the message m according to the template constraints. This measure restricts the semantics of the signed information, counteracting certain attacks. However, and as previously mentioned, this validation is intended to evaluate the security measures against DY model. For that reason, the template retrieval by the origin in the first step of the main protocol was discarded as well.

Server role capabilities

Finally, the *server* role (the TTP) was initially too complex for the backend analyzers of AVISPA, due to the number of transition combinations considered during the validation process. Note that our TTP is designed as an e-notary, storing and managing the evidences generated during a protocol run in which the TTP takes part. Besides, there were three situations where the TTP could intervene: an abort requested by either e_1 or e_2, and a recovery requested by the receiver. This led to a huge role definition with 9 transitions. Taking into account that CL-AtSe considers, by default, that each transition can be applied at most 3 times, the backend had to manage 27 possible transitions during the analysis. It seemed to be too complex.

For that reason, we simplified the role server, excluding e_2 from making abort requests. The origin is still able to abort the protocol by using e_1. Thus, the server role was modeled considering next three states: 0 as the initial state; 1 as the state when an abort has been done first; and 2 as the state when a recovery has been done first. In each state, the server can receive both abort and recovery requests, leading to 6 defined transitions.

As a result of previous modifications applied, the server role cannot respond to several parallel sessions. The transitions are sequential, that is, once a transaction has been aborted or recovered, the server will stay in that state (1 or 2, respectively) for the rest of requests, no matter if they come from another session. Nonetheless, and as will be seen further, we were able to test the protocol with parallel sessions between e_1 and e_2 and *receiver*, looking for possible attacks though the server behaved in this way.

We know that we have limited the possible space of attacks, but the decision was made in order to allow the backends to correctly analyze the protocol.

10.4.1.2 Analysis scenarios

HLPSL must cover the definition of two special roles: session and environment. Respecting the session role, we just instantiated the roles e_1, e_2, and *receiver* with the adequate information. Mention that the agents e_1 and e_2, since both are managed by the same origin, own a pre-shared knowledge: the message to be sent. The template reference is also a pre-shared knowledge between e_1, e_2 and the *receiver* agents.

One of the most important parts of the HLPSL is the initial knowledge allocated to the intruder. In this sense, and for test purposes, we defined five different template references (tpl_id, tpl_id2, tpl_id3, tpl_id4) and four different messages (msg, msg2, msg3, msg4), assigning the subset {msg2, msg3, tpl_id2, tpl_id3, tpl_id4} as knowledge to the intruder.

Afterwards, we defined several analysis scenarios with different sessions configurations (see Section 10.4.3 for details). Due to the constraints applied to the server role capabilities commented above, we instantiated just one server role in each scenario.

Security goals

AVISPA supports three types of goals so far: *secrecy_of*, (strong) *authentication_on* and *weak_authentication_on*. In the latter, the piece of data used to authenticate the agent can be reused by an attacker, so reply attacks are not considered by the analyzers. As AVISPA does not explicitly support fairness and non-repudiation security goals, some fair exchange/non-repudiation protocols modeled with AVISPA used *secrecy_of* goal to achieve it. However, OFEPSP+ does not provide confidentiality on any item. Fairness is achieved by means of the signature policies fulfillment. Therefore, currently we have only modeled authentication goal respecting the exchanged evidences.

10.4.2 HLPSL correctness verification

In this step, the aim was twofold: verify the syntactic/semantic correctness and executability of the HLPSL specification; and that the HLPSL specification implemented the intended protocol behavior. For the syntactic, semantic and executability verification, next AVISPA and SPAN tools were used:

HLPSL2IF: This tool translates the HLPSL specification into the Intermediate Format (IF). A successful translation implies that the syntax of the protocol is correct. OFEPSP+ HLPSL file was correctly translated into the corresponding IF.

Protocol simulation: By simulating the protocol with SPAN, the semantic of the protocol's HLPSL is verified and a Message Sequence Chart (MSC) visualized. In our case, the semantic was correctly verified but the MSC could not be shown. It frequently happens when the transition labels and state values are not perfectly set. In any case, it does not imply an error in the specification.

OFMC search tree option: OFMC offers the possibility to browse the search tree through a path indicated by the indexes of the successors to follow. As a result, one can decide which choice point take in a specific point of the search tree deduced from the IF. This option allows the tester to check if every transition can be taken during a protocol run. In our case, every transition could be chosen at some time in the search tree.

CL-AtSe no executability option: CL-AtSe offers the possibility of tracing the protocol specification without being analyzed. The output shows the so called *Initial System State*, representing both the intruder and honest participant's states in CL-AtSe just after reading and interpreting the IF file. While the intruder state is just represented by a list of knowledges, the honest participants are described by a set of instantiated roles, called *Interpreted protocol specification.* This option is useful to check that CL-AtSe interprets the protocol transitions as expected. Each role consists in a tree where unary nodes are protocol steps and n-ary nodes are choice points. In our case, each possible transition was represented in the tree.

SATMC check only executability: With this option, SATMC checks on executability of actions/rules without any intruder, allowing the tester to debug the specification. The output trace showed that every rule could be executed.

Session compilation with OFMC: With session compilation (sessco), OFMC finds a replay attack even without a second parallel session. It first simulates a run of the whole system and in a second run, it lets the intruder take advantage of the knowledge learned in the first run. Sessco is also handy for a quick check of executability. However, as stated in AVISPA documentation, if one role can loop (i.e. remain in the same control state forever and make infinitely many steps), sessco is not possible, and OFMC aborts with an error message. That is our case in some transitions of role server, and thus, we could not use this option.

CL-AtSe no executability option and *OFMC search tree option* helped us also to ascertain that the HLPSL specification matched the intended protocol behavior.

10.4.3 OFEPSP+ security validation

The results obtained from validating OFEPSP+ with OFMC and CL-AtSe backends are summarized in next Tables 10.3, 10.4 and 10.5. In case of SATMC, the result was always "Inconclusive". Tests reports showed us that SATMC did not find an attack, but it warned that, with SATMC backend, intruder is not allowed to generate fresh terms (i.e. e_1 role). As a consequence, attacks based on such an ability would not be reported. TA4SP was not used because it does not support *sets* up to now. The analysis scenarios referred in these Tables are described in Table 10.6[1].

Analysis scenario	OFMC	CL-AtSe
cfg1	SAFE	SAFE
cfg2	SAFE	SAFE
cfg3	SAFE	SAFE
cfg4	SAFE	SAFE

Table 10.3: Validation results with OFMC and CL-AtSe respecting a single session with legitimate agents and single sessions with intruder playing the role of a legitimate agent

Configurations applied in Table 10.3 were aimed at finding attacks in a normal session (cfg1) or sessions where the intruder impersonates one of the legitimate agents - E_1 (cfg2), E_2 (cfg3) or *Receiver* (cfg4).

Analysis scenario	OFMC	CL-AtSe
cfg5	SAFE	SAFE(*)
cfg6	SAFE	SAFE
cfg7	SAFE	SAFE
cfg8	SAFE	SAFE

Table 10.4: Validation results with OFMC and CL-AtSe respecting parallel sessions with legitimate agents playing different roles

[1]Intruder is denoted as 'i'. We did not instantiated any session with the intruder playing the role of the server because we consider the TTP to be honest.

Analysis scenario	OFMC	CL-AtSe
cfg9	SAFE(*)	SAFE
cfg10	SAFE(*)	SAFE
cfg11	SAFE(*)	SAFE
cfg12	SAFE	SAFE
cfg13	SAFE	SAFE
cfg14	SAFE	SAFE

Table 10.5: Validation results with OFMC and CL-AtSe respecting parallel sessions with intruder playing as legitimate agent(s)

Configurations shown in Table 10.4[1] were focused on violating the security goals when two coherent parallel sessions are executed (cfg5) and when a legitimate party is playing a role for which is not intended to in case of parallel sessions (cfg6, cfg7 and cfg8). The difference between these two types of scenarios lies in that, in the latter, a legitimate agent poses as a different one. For instance, in one session, each agent is playing the corresponding role, while in the second session, the origin's environments play the role of the other environment (cfg8). The backend analyzer uses the information simultaneously generated in both session to mix the messages between the sessions and see if an attack on the security goals can be executed. We realized that each participant's identifier had to be included in each evidence generated in order to avoid this sort of attack. In particular, we used the public keys of E_1, E_2, *Receiver*, and *TTP*.

Table 10.5[2] contains the set of configurations where an intruder is playing the role of legitimate agent(s) when two parallel sessions are executed. Note that the knowledge own by the intruder in each configuration differs. The aim was to find possible security goals violations in a session when carried out from an intruder running in another different session. Mention that when the intruder plays a legitimate role in a session, the goals involving him are not considered by AVISPA (otherwise, he could always achieve an attack).

Based on the results obtained from the tests, our protocol fulfills the security goals G2 - Message authentication (includes message integrity), G17 - Accountability, G18 - Proof of Origin and G19 - Proof of Delivery for partial evidence $PNRO_1$, $PNRR_1$, $PNRO_2$, $PNRR_2$, and valid evidence NRE and NRE_{TTP}. The description of goals can be found in Deliverable 6.1 "List of selected problems" in AVISPA project [15].

Due to our protocol design, evidences are not protected against reply attacks (G3), and thus goal Entity Authentication (G1) is not achieved either. For that reason,

[1]OFMC executed with maximum search depth of 17. CL-AtSe executed with -nb 1 option (maximum 1 loop iteration in any trace) for test marked with (*)

[2]Results marked with (*) mean that OFMC was launched with maximum search depth of 17

Analysis scenario	sessions configuration
cfg1	$session\,(e_1, e_2, r, s, ..., msg, tpl\_id)$
cfg2	$session\,(i, e_2, r, s, ..., msg, tpl\_id)$
cfg3	$session\,(e_1, i, r, s, ..., msg, tpl\_id)$
cfg4	$session\,(e_1, e_2, i, s, ..., msg, tpl\_id)$
cfg5	$session\,(e_1, e_2, r, s, ..., msg, tpl\_id)$ $session\,(e_1, e_2, r, s, ..., msg, tpl\_id)$
cfg6	$session\,(e_1, e_2, r, s, \ldots, msg, tpl\_id)$ $session\,(r, e_1, e_2, s, \ldots, msg, tpl\_id)$
cfg7	$session\,(e_1, e_2, r, s, \ldots, msg, tpl\_id)$ $session\,(e_1, r, e_2, s, \ldots, msg, tpl\_id)$
cfg8	$session\,(e_1, e_2, r, s, \ldots, msg, tpl\_id)$ $session\,(e_2, e_1, r, s, \ldots, msg, tpl\_id)$
cfg9	$session\,(e_1, e_2, r, s, \ldots, msg, tpl\_id)$ $session\,(i, e_2, r, s, \ldots, msg, tpl\_id)$
cfg10	$session\,(e_1, e_2, r, s, \ldots, msg, tpl\_id)$ $session\,(e_1, i, r, s, \ldots, msg3, tpl\_id3)$
cfg11	$session\,(e_1, e_2, r, s, \ldots, msg, tpl\_id)$ $session\,(e_1, e_2, i, s, \ldots, msg4, tpl\_id4)$
cfg12	$session\,(i, e_2, r, s, \ldots, msg2, tpl\_id2)$ $session\,(e_1, i, r, s, \ldots, msg3, tpl\_id3)$
cfg13	$session\,(i, e_2, r, s, \ldots, msg2, tpl\_id2)$ $session\,(e_1, e_2, i, s, \ldots, msg3, tpl\_id3)$
cfg14	$session\,(e_1, i, r, s, \ldots, msg, tpl\_id)$ $session\,(e_1, e_2, i, s, \ldots, msg, tpl\_id)$

Table 10.6: Analysis scenario configurations for the tests

only *weak_authentication_on* goal was assigned. We think that including a nonce in the requests generated by the receiver would enforce goals G1 and G3 for $PNRO_2$, $PNRR_2$, NRE and NRE_{TTP}. However, tests conducted including that nonce did not reach a conclusion, maybe due to the huge number of transitions the backends had to analyze. As a result, our assumption could not be formally proved.

10.4.4 Informal analysis of fairness property

This Section intends to fill the gap left by the formal validation regarding the analysis of OFEPSP+ fairness security goal. For that purpose, we perform an informal analysis of the security of OFEPSP+, focusing on the expected fairness property of the protocol.

In our analysis, the goal of the attacker is to obtain a valid evidence from the other side without making any commitment in the transaction. The achievement of this goal would imply that the fairness of the protocol has been broken. There are mainly four possible attacks, which are based on protocol interruptions. This kind of attack consists of aborting the protocol in a chosen step. It should be noted that sometimes a communication or system error can cause the same effect:

- *Main protocol interruption after $PNRO_1$ reception.* The receiver stops the protocol just after step (3), causing the origin not obtaining any evidence from the receiver. However, due to the conditions established in the extended electronic signature policy, that dictate that only a set of signatures complying with NRE or NRE_{TTP} composition are the valid evidence, the receiver could not make use of $PNRO_1$ as a valid one.

- *Main protocol interruption after $PNRR_1$ reception.* The origin stops the protocol just after step (5) and after having received the $PNRR_1$. Therefore, the receiver possesses the $PNRO_1$ and the origin the $PNRR_1$. For the same reason as before, this evidence cannot be considered as complete.

- *Main protocol interruption after $PNRO_2$ reception.* The receiver stops the protocol just after step (7), and after having received the $PNRO_2$. Again, this evidence cannot be considered as complete. But, in this case, the receiver can generate $PNRR_2$ and execute the recovery subprotocol. If the origin has not aborted the protocol before, the receiver will obtain NRE_{TTP} from the TTP. However, the origin is expected to abort the protocol at some time, obtaining the NRE_{TTP} as well.

- *Main protocol interruption after $PNRR_2$ reception.* The origin stops the protocol just after step (8), and after having received the $PNRR_2$. In this case, the origin is capable of producing valid evidence NRE. If the origin aborts the protocol immediately after step (8), the receiver will not be able to obtain valid evidence from the TTP, while the origin did. This situation is unfair to the receiver. However, and as discussed in Section 9.1.6, the origin's misbehavior would be uncovered during a further dispute resolution. As a result, though this situation is unfair to the receiver at first instance, it is finally detrimental to the malicious origin. It should be remarked that, if the origin aborts the protocol before receiving $PNRR_2$, he will not be able to compose valid evidence, as the participation of the receiver is needed.

As can be observed, every potential attack is counteracted. As a result, we can conclude that fairness is guaranteed in OFEPSP+.

10.5 Chapter Summary

This Chapter has presented the suite of evaluations performed to prove the improvements, correctness, feasibility and security of the main contributions made in this thesis.

The taxonomy has been evaluated against the set of general requirements for taxonomies, showing that our taxonomy complies with most of them. Notwithstanding, our taxonomy requires expert security knowledge for its understanding and application, and therefore it does not comply with the *comprehensible* requirement. In addition, we do not guarantee a deterministic classification of attacks, since we consider that it is not possible in this inexact field of study. In this sense, the proposed method of classification intends to reduce the ambiguity during the classification procedure, but the final result will strongly depend on the training, skills and perspective of the user in charge of the classification and the available information of the attack.

The evaluation has also covered the analysis of the enhancement of the reliability of digital signature-based evidence achieved by the thesis proposal. Formal proofs have been given, and demonstrate that the division of the signature environment is an effective approach that can substantially decrease the probability of a successful attack. Contrary to other proposals, the division principle does not rely on the existence of a trusted terminal or device. Moreover, the number of environments can be balanced depending on the required degree of reliability to achieve. Though from a theoretical perspective, our paradigm is the only one that permits achieving perfect security in the sense that the probability of a successful attack can be reduced to zero.

In addition, the theoretical analysis has been extended to evaluate the improvement achieved by our thesis proposal respecting each attack method identified in the taxonomy. The results demonstrate that our proposal improves current state-of-the-art while considering a holistic threat model.

An experimental implementation of the validation algorithm of the extended electronic signature policy framework and a simulator for OFEPSP+ have been developed in order to prove the correctness of the design and the feasibility for a practical implementation. This implementation has also served to prove the feasibility of the division paradigm, as it has been virtually applied in the simulator to the origin of OFEPSP+ protocol.

Finally, automated reasoning techniques have been used by means of AVISPA and SPAN tools in order to formally verify the security of OFEPSP+, taking the intruder model of Dolev and Yao as the reference threat model. The results of the security analysis show that our protocol assures authentication and integrity security goals. Due to the limitation found in the tools, the formal validation has been complemented with an successful informal analysis of OFEPSP+ fairness property.

Chapter 11

Conclusions and Future Work

This Chapter contains the thesis conclusions and final remarks, and summarizes the contributions achieved. Additionally, future research directions that derive from the thesis results are proposed.

11.1 Conclusions

The research undertaken in this thesis has been focused, in the first instance, on the study and formalization of the security threats that may subvert the reliability of digital signature-based evidence, and, in the second instance, on devising a new proposal that better counteracted those threats. The pursued final goal was to significantly enhance the reliability of digital signatures, enforcing their non-repudiation property when acting as evidence.

The reliability of a digital signature should determine its capability to be used as valid evidence. In this sense, this thesis contributes to obtain higher assurance respecting the actions or events occurred in electronic transactions, and attested by digital evidence. The same applies when digital signatures are used as instrument of evidence, like in e-commerce or transactions where digital signatures play an important role, and have legal effectiveness. A party will always be able to (try to) repudiate the commitment made in a signed document or the action or event attributed in a certain transaction. However, with our proposal, the judge or the party in charge of resolving the dispute will possess a more reliable evidence, being capable to make an informed decision that will be surely closer to what actually happened.

In addition, contrary to what stakeholders and legislators expected, digital signatures have not spread as fast as desired. Apart from economical and political reasons, studied by Berta in [26], the most critical influential factor is the lack of trust in technology, specially when one's interests and money are put in place.

A well known definition of the concept of trust, adjusted to the case of two parties involved in a transaction, is that *an entity A is considered to trust another entity B when entity A believes that entity B will behave exactly as expected and required.* The same applies between the relationship established between end users and technology. Trust in the information society is built from several circumstances, like calculus, knowledge, social reasons or experience [151]. As a result, being aware of existent security threats, having suffered bad experiences when using technology or just being reluctant to use technology because of a feeling of lack of control regarding what is actually happening will inevitably damage the trust in technology. Consequently, we consider that the contributions of this thesis will help increasing the users' confidence and trust in electronic transactions, once the control of their own information and actions is clearly strengthened.

Next, the specific contributions achieved in the thesis are listed, along with the scientific papers published and patents filed.

- Formal and holistic study of the security threats on digital signatures [97, 99]

  A **taxonomy of attacks** has been devised in Chapter 6, and which categorizes the attacks that can be carried out on the digital signature creation and verification stages. A method for the systematic classification of attacks has also been provided, establishing the steps that must be followed to classify an attack under the dimensions of the taxonomy.

  The taxonomy has been successfully evaluated (Section 10.1), and a **comprehensive survey and classification of attacks** using our taxonomy has also been performed (Appendix B). The survey proves to some extent that current technology is not capable of counteracting existent security threats, and thus fails to offer a resilient solution for digital signatures.

- Proposal to enhance the reliability of digital signature-based evidence [93, 96, 100, 101]

  The proposal consists of two main pillars: the **signature environment division paradigm**, formalized in a technology-independent approach in Chapter 7, and the **extended electronic signature policy**, defined in Chapter 8.

  The formal proofs that demonstrate the benefits of the environment division and the improvement against the current state of the art have been given in Section 10.2. Also, the security mechanisms for the generation and verification of digital signature-based evidence have been detailed. These mechanisms include two different schemes: the chaining mode scheme and the independent mode scheme. The first scheme is recommended when the order of the environments being used is important, while in the second one the generation and verification processes does not need to follow a certain order.

Another important issue regarding the implementation of the division principle is the binding between the environment and the signature creation data. This aspect has also been studied in Chapter 7, proposing two different approaches to resolve it: binding by procedure and binding by environment attestation. It is also worth mentioning the remark given in this Chapter respecting the format of the data to be signed. We concluded that it cannot be guaranteed the integrity of the semantic of some data if a complex or rich format is used. This is important since the format of the potential data to be signed should be restricted to static file formats in order to counteract certain threats identified in the taxonomy.

On the other hand, the extended signature policy has been formally described in ASN.1 and XML, along with the generation and validation procedures. This policy supports the definition of the dependences and relationships among the signatures generated in the same transaction. As a result, the policy permits the practical implementation of the division paradigm, stipulating the signatures that are required to satisfy the evidence validity.

The pseudo-code of the tree matching and validation algorithm has also been provided in Appendix C, while the details of the implementation and evaluation of the validation process have been included in Section 10.3.

The security of the division paradigm supported by the extended signature policies has been evaluated in Section 10.2 against the categories of attacks identified in the taxonomy, analyzing to what extent our proposal is resilient against such threats. The results demonstrate that our proposal substantially improves the current state of the art. However, we know that (practical) perfect security does not exist, and thus realistic research efforts must be oriented to enhance the security of current systems to a greater or lesser extent.

- An optimistic fair exchange protocol based on the signature environment division paradigm and extended signature policies [94, 95, 96, 98]

  The thesis proposal has been put into practice with the design and implementation of a practical use case. In particular, a **new fair exchange protocol** has been proposed in Chapter 9 using a design based on two environments and an extended electronic signature policy, which ASN.1 definition has been included in Appendix F. Implementation guidelines, like the architecture for the signature policies management, assumptions on the communication channels or the electronic signature formats to use have also been explained. In addition, the protocol steps have been generalized to permit the incorporation of as many environments as desired.

The security of the protocol has been formally validated using automated reasoning technique tools. Due to the limitations of the formal validation, a complementary informal validation of the fairness property has been performed. Appendix G contains the protocol specification used for the analysis, in High-Level Protocol Specification Language, while the results of the analysis itself have been given in Section 10.4.

Finally, the protocol has been implemented and tested in a simulation (see Section 10.3).

11.2 Future Work

The contributions achieved and the evaluations performed on the thesis results are promising, and confirm that we have improved the current state of the art. However, some work must still be done to complete a working prototype and fill some needs identified herein. In addition, future research directions arise, and which investigation would be of high interest. These issues are commented next:

- Formalize an attack model proposal that could be integrated in the taxonomy

  An attack model is used to define how attacks are performed. Modeling attacks is useful to design appropriate countermeasures to mitigate them, that is, reduce the probability of occurrence of an attack by eliminating the vulnerabilities exploited. On the other hand, a taxonomy of attacks is intended to permit the classification of attacks for its further understanding, and thus design effective countermeasures.

  Integrating an attack modeling phase during the classification of an attack would significantly increase the accuracy of the attack behavior, and consequently the classification itself. The user would model the attack, decomposing the attack behavior into its atomic subattacks. Afterwards, the user would use the taxonomy to classify the attack taking into account the detailed information of the model.

  However, the modeling phase is complex on its own. In addition, blended attacks are not easily classifiable under current taxonomies. As the attack modeling strongly depends on the information available of the attack, an attack modeled for its later classification could make a deterministic classification even more difficult. If two models of the same attack differ, then the classification result will inevitably differ as well.

To minimize this risk, it would be advisable to develop a detailed set of modeling instructions in a manner that the user of the taxonomy is guided through a well-defined and constrained path. The instructions should be capable of establishing in a clear manner what implies each level of abstraction in the model, how the attack or subattacks are decomposed into more refined processes and what information of each resultant subattack is used for the identification of the categories of each dimension.

- Integrate security metrics for the adjustment of the number of environments

  The signature environment division paradigm explained in Chapter 7 relies on the lower probability of a successful attack (PSA) achieved when more than one signature environment is used. This PSA is the result of the PSAs of each environment. Though some concrete examples have been described, these PSAs are taken as abstract figures which actual value do not care for the rationale, as the resultant combined probability is always lower independently of each single value. However, having concise information about such probabilities would allow making better decisions when choosing the environments. For instance, if one knew the actual lowest PSA among different environments, then the corresponding environment should be selected in a homogeneous set of environments.

  In our opinion, calculating the exact PSA is a very difficult task, specially when the environment to measure is under the control of a domestic end user. However, there are some research areas which objective is precisely to devise ways to measure security related factors and thus permit affected parties to react accordingly.

  As commented by Jansen in [125], information security metrics are an important factor in making sound decisions about various aspects of security, offering a quantitative and objective basis for security assurance. Measurements provide single-point-in-time views of specific, discrete factors, while metrics are derived by comparing to a predetermined baseline two or more measurements taken over time [182].

  Next, a good definition extracted from [222], and that illustrates how security metrics and measurements can be used to estimate the PSA of certain environment, is given:

  "*At a high-level, metrics are quantifiable measurements of some aspect of a system or enterprise. For an entity (system, product, or other) for which security is a meaningful concept, there are some identifiable attributes that collectively characterize the security of that entity. Further, a security metric (or combination of* *security metrics) is a quantitative measure of how much of that attribute the entity possesses. A security metric can be built from lower-level physical measures*".

Standardized security evaluation methodologies, like Common Criteria [54], reviewed in Chapter 5, are an example of qualitative measures used by an evaluator to estimate the security of a product or system. However, and due to the inevitable subjectivity involved, these measures are usually non-repeatable by other evaluators. Moreover, obtaining objective quantitative measures from which calculating the PSA of certain environment may be even more difficult, since measurements of software properties in general has been difficult to accomplish [125].

In [125], the author provides a list of possible research areas in security metrics and measurements. The division paradigm proposed in this thesis can benefit from the achievements of these challenging research efforts.

- Detail the environment attestation technique

  Chapter 7 explained how the division paradigm could be implemented, including the evidence generation and verification schemes and two proposals to bind each environment from which evidence is generated with the corresponding signature creation data. With this respect, the environment attestation technique was the proposal that permitted a better way of performing such binding. However, the information provided merely outlined how this proposal could be implemented.

  We see that it is necessary to go into the environment attestation in depth. In our opinion, the elaboration of an approach based on Trusted Platform Module (TPM) or Mobile Trusted Module (MTM) would be the next step in the implementation specifications.

- Support for multiple signed data

  Current extended electronic signature policy assumes that every primary signature is generated on the same piece of information. The result led to a tree graph definition where the root node was considered as the signed data while the first level of nodes were the primary signatures, either parallel or sequential. The fair exchange protocol designed complies with this assumption as the origin and receiver generate their corresponding first partial evidence on the same information, that is, the message sent by the origin to the receiver.

  However, a transaction may imply that more than one message, each of which to be attested by digital evidence, is exchanged between the parties involved. In this case, and taking into account the extended policy definition, a different extended policy would have to be used for each message exchanged, and where each policy established the rules for the signatures to be generated and verified respecting the corresponding message. This would make that the link between the signatures, when defined in different policies, was lost.

We think that it would be interesting to evolve the extended policy definition to support signatures over multiple data that belonged to the same transaction. For instance, an identifier could be added to each tree of signatures defined in the tree of solutions of the extended policy. Trees with the same identifier would be conceptually bound to the same piece of information. One tree of signatures with each different identifier should have to be fulfilled in order to make the transaction effective.

It would still be needed to restrict the particular message, among all messages of the transaction, on which certain set of signatures can be generated, as well as defining the acceptable combinations between trees with different identifiers, and the relationships between signatures that belong to different trees. For example, let's assume a fair exchange protocol where two trees of signatures, with a different identifier each, must be fulfilled. One tree over the message sent by the origin and another tree over the receipt sent by the receiver. Let's also assume that there are recovery and abort subprotocols associated to both the message and the receipt. It would not be coherent if the tree of signatures that represents the recovery subprotocol of the message was fulfilled at the same time as the tree bound to the abort subprotocol of the receipt. Likewise, it may be required that the signatures corresponding to the abort subprotocol of the origin's message are generated at a time before those corresponding to the abort subprotocol of the receipt.

- Support dynamic establishment of absolute timing and sequence dependence

  Absolute timing and sequence constraints for some signature is currently defined in the extended policy by setting the actual dates for the time period. However, it is possibly not practical as the policy issuer may not know, a priori, at what time each signature will be produced.

  We think that the procedures defined in Chapter 8 should be enhanced in order to permit the dynamic establishment of any time period of the absolute timing and sequence dependences represented in the policy. This approach has the drawback of requiring that the policy is issued for each transaction, possibly affecting the performance. It would be advisable to research other alternatives.

- Implement graphical tools

  Several parts of the proposal imply complex processes which completion would be facilitated if graphical tools were used. For example, the visualization of tree information during the generation and validation of multi signature-based evidence according to an extended signature policy. In particular, in the second step during the signature generation process (*policy information visualization and pre-processing*, see Section 8.2.1) the signer has to evaluate the extended signature policy information and the partial set of signatures, if available. This information should be shown in a format as usable as possible, possibly using charts.

- Extend the Dolev-Yao threat model to support non-perfect cryptography

  Dolev and Yao threat model, used in AVISPA and SPAN for the formal validation of security protocols, assumes perfect cryptography. The intruder cannot break cryptography. However, the taxonomy of attacks proposed in this thesis has clearly shown cryptanalysis as a potential security threat to digital signatures.

  In [39], the authors propose an approach to weaken this hypothesis, by means of probabilistic considerations on the strength of cryptographic functions. A natural extension of the threat model used to formally validate the security of OFEPSP+ would be to integrate this proposal, being capable of considering more realistic scenarios.

- Formally validate fairness in OFEPSP+

  As commented in the evaluation of OFEPSP+, Section 10.4, AVISPA does not explicitly support fairness and non-repudiation security goals. In [67], it is demonstrated that the fair exchange security goal can be reduced, via a meta-reasoning step, to a secrecy goal. Thereby, fair exchange protocols that apply symmetric encryption to provide fairness are able to use *secrecy_of* goal to validate fairness. However, OFEPSP+ does not provide confidentiality on any item. Fairness is achieved by means of the signature policies fulfillment, and as result, this approach could not be used.

  Another recent approach to validate fairness consists of using special predicates (*has*, *none*, *aknows*) and goal formulas [136]. In this case, only CL-AtSe backend support them. It would be recommended to complete OFEPSP+ formal validation by using this second approach.

Part V

Bibliography and Appendices

Bibliography

[1] O. AciiÇmez. Yet another MicroArchitectural Attack: Exploiting I-cache. Conference on Computer and Communications Security. ACM workshop on Computer Security Architecture, pp. 11–18 (2007) 269

[2] O. AciiÇmez, Ç. K. KoÇ, and J.-P. Seifert. On The Power of Simple Branch Prediction Analysis. ACM Symposium on Information, Computer and Communications Security (ASIACCS'07), pp. 312–320, ACM Press (2007) 269

[3] O. AciiÇmez, Ç. K. KoÇ, and J.-P. Seifert. Predicting Secret Keys via Branch Prediction. Topics in Cryptology - RSA Conference 2007, LNCS 4377, pp. 225–242, Springer–Verlag (2007) 267, 268

[4] O. AciiÇmez, W. Schindler, and Ç. K. KoÇ. Improving Brumley and Boneh Timing Attack on Unprotected SSL Implementations. 12th ACM Conference on Computer and Communications Security, pp. 139–146, ACM Press (2005) 263

[5] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Internet X.509 Public Key Infrastructure. Time-Stamp Protocol (TSP). Internet Engineering Task Force – Request for Comments 3161 (2001) 151, 191

[6] C. Adams, S. Farrell, T. Kause, and T. Mononen. Internet X.509 Public Key Infrastructure. Certificate Management Protocol (CMP). Internet Engineering Task Force – Request for Comments 4210 (2005) 24, 53, 283

[7] J. M. Alonso, R. Bordon, M. Beltran, and A. Guzman. LDAP injection techniques. 11th IEEE Singapore International Conference on Communication Systems (ICCS 2008), pp. 980 – 986 (2008) 286

[8] A. Alsaid and C. J. Mitchel. Dynamic content attacks on digital signatures. Information Management & Computer Security 13(4). pp. 328–336 (2005) 51, 63

[9] E. Amoroso. Fundamentals of Computer Security Technology. Prentice-Hall (1994) 167

[10] Annual Report. Panda Labs (2009) 7, 81

[11] N. Asokan. Fairness in electronic commerce. PhD thesis, Waterloo, Ontario, Canada (1998) 38

[12] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. 4th ACM conference on computer and communications security, pp. 7–17 (1997) 39

[13] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Heam, O. Kouchnarenko, J. Mantovani, S. Modersheim, D. von Oheimb, M. Rusinowitch, J. Santos Santiago, M. Turuani, L. Vigano, and L. Vigneron. The AVISPA Tool for the automated validation of internet security protocols and applications. 17th International Conference on Computer Aided Verification (CAV'2005), LNCS 3576, pp. 281–285, Springer (2005) 199

[14] A. Armando and L. Compagna. SATMC: a SAT-based model checker for security protocols. 9th European Conference on Logics in Artificial Intelligence (JELIA'04), LNAI 3229, pp- 730–733, Springer–Verlag (2004) 199

[15] AVISPA: Automated validation of internet security protocols and applications (2003) FET Open Project IST-2001-39252. `http://www.avispa-project.org/` 199, 205

[16] AVISPA. Deliverable 2.1: The High-Level Protocol Specification Language (2003). Available at `http://www.avispa-project.org/publications.html` 199

[17] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing, 1(1), pp. 11–33 (2004) 67, 79

[18] A. Axelsson. Intrusion Detection Systems: a Survey and Taxonomy. Technical Report No 99-15, Department of Computer Engineering, Chalmers University, Gothenburg (2000) 42

[19] B. Balacheff, D. Chan, L. Chen, S. Pearson, and G. Proudler. Securing Intelligent Adjuncts Using Trusted Computing Platform Technology. IFIP TC8/WG 8.8 4th Working Conference on Smart Card Research and Advanced Applications, pp. 177–195 (2000) 61

[20] B. Balacheff, L. Chen, S. Pearson, D. Plaquin, and G. Proudler. Trusted Computing Platforms: TCPA Technology in Context, Prentice Hall PTR, Upper Saddle River, New Jersey (2003) 62

[21] B. Balacheff, L. Chen, D. Plaquin, and G. Proudler. A trusted process to digitally sign a document. Workshop on new security paradigms, pp. 79–86 (2001) 61

[22] D. Balfanz and E. W. Felten. Hand-held computers can be better smart cards. 8th USENIX Security Symposium, pp. 15–24 (1999) 58

[23] F. Bao, R. H. Deng, and W. Mao. Efficient and practical fair exchange protocols with off-line TTP. IEEE symposium on security and privacy (1998) 38

[24] E. Barker. Recommendation for Obtaining Assurances for Digital Signature Applications. NIST Special Publication 800–89 (2006) 51

[25] D.A. Basin, M. Sebastian, and L. Vigano. OFMC: A symbolic model checker for security protocols. International Journal of Information Security, 4(3), pp. 181–208 (2005) 199

[26] I. Z. Berta. Why are not digital signatures spreading as quickly as it was expected? MBA dissertation, Buckinghamshire Chilterns University College, Buckinghamshire Business School, Számalk Open Business School (2004) 209

[27] I. Z. Berta. Mitigating the attacks of malicious terminals. PhD Dissertation. Budapest University of Technology and Economics (2005) 55, 61

[28] I. Z. Berta. Using multiple smart cards for signing messages at malicious terminals. 9th Information Security Conference. LNCS 4176, pp. 246–256 (2006) 56

[29] I. Z. Berta, L. Buttyán, and I. Vajda. Mitigating the Untrusted Terminal Problem Using Conditional Signatures. International Conference on Information Technology (ITCC 2004) (2004) 55

[30] I. Z. Berta, L. Buttyán, and I. Vajda. A Framework for the Revocation of Unintended Digital Signatures Initiated by Malicious Terminals. IEEE Transactions On Dependable And Secure Computing, 2(3), pp. 268-272 (2005) 55

[31] I. Z. Berta and I. Vajda. Limitations of humans when using malicious terminals. Tatra Mountains Mathematical Publications (2005) 63

[32] R. Bevan and E. Knudsen. Ways to Enhance DPA. ICISC 2002, LNCS 2587, pp. 327–342, Springer–Verlag (2003) 264

[33] K. Bicakcia and N. Baykalb. Improved server assisted signatures. Computer Networks, 47(3), pp. 351–366 (2005) 62

[34] M. Bishop. A taxonomy of (Unix) system and network vulnerabilities. Technical Report CSE–9510. Department of Computer Science, University of California (1995) 42

[35] M. Bishop. Vulnerabilities Analysis. International Symposium on Recent Advances in Intrusion Detection (1999) 42

[36] M. Bishop and D. Bailey. A critical analysis of vulnerability taxonomies. Technical Report CSE–96–11. Department of Computer Science, University of California (1996) 42, 43

[37] Y. Boichut, P.-C. Heam, O. Kouchnarenko, and F. Oehl. Improvements on the Genet and Klay Technique to Automatically Verify Security Protocols. Automated Verification of Infinite States Systems (AVIS'04), pp. 1–11 (2004) 199

[38] S. Bratus, N. D'Cunha1, E. Sparks, and S. W. Smith. TOCTOU, Traps, and Trusted Computing. LNCS 4968, pp. 14–32 (2008) 62

[39] R. Bresciani and A. Butterfield. Weakening the Dolev-Yao model through probability. 2nd ACM International Conference on Security of Information and Networks (SIN 2009), pp. 293–297, ACM Press (2009) 216

[40] M. A. Broderick, V. R. Gibson, and P. Tarasewich. Electronic Signatures: They're legal, Now What? Internet Research: Networking Applications and Policy, 11(5), pp. 423–434 (2001) 29

[41] D. Brumley and D. Boneh. Remote Timing Attacks are Practical. 12th Usenix Security Symposium, pp. 1–14 (2003) 56, 262

[42] F. Buccafurri and G. Lax. Hardening digital signatures against untrusted signature software. 2nd International Conference on Digital Information Management (ICDIM '07), pp. 159-164 (2007) 64

[43] F. Buccafurri, G. Caminiti, and G. Lax. The Dalì Attack on Digital Signature. Journal of information assurance and security. Dynamic Publishers Inc., USA (2008) 244, 279

[44] F. Buccafurri, G. Caminiti, and G. Lax. Fortifying the Dali Attack on Digital Signature. 2nd ACM International Conference on Security of Information and Networks (SIN 2009), pp. 278–287, . ACM Press (2009) 244, 279

[45] CEN Workshop Agreement CWA 14169 – Secure signature–creation devices EAL 4+. The European Committee for Standardization (CEN) (2004) 50, 70

[46] CEN Workshop Agreement CWA 14170 – Security Requirements for signature creation applications. The European Committee for Standardization (CEN) (2004) vii, 9, 43, 51, 69, 70, 72, 167

[47] CEN Workshop Agreement CWA 14171 – General guidelines for electronic signature verification. The European Committee for Standardization (CEN) (2004) vii, 9, 43, 51, 72, 73, 74, 90, 167, 284

[48] A. Chari, J. Rao, and P. Rohatgi. Template Attacks. 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES02), LNCS 2523, pp. 51–62, Springer–Verlag (2002) 265

[49] Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, J. Mantovani, S. Modersheim, and L. Vigneron. A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols. Workshop on Specification and Automated Processing of Security Requirements (SAPS 2004), Austrian Computer Society (2004) 199

[50] S. Chokhani, W. Ford, R. Sabett, C. Merrill, and S. Wu. Internet X.509 Public Key Infrastructure. Certificate Policy and Certification Practices Framework. Internet Engineering Task Force – Request for Comments 3647 (2003) 22, 54, 284

[51] T. Coffey and P. Saidha. Non-repudiation with mandatory proof of receipt. ACM SIGCOMM (1996) 39

[52] Common Attack Pattern Enumeration and Classification (CAPEC)- A Community Knowledge Resource for Building Secure Software. MITRE Corporation. Available at `http://capec.mitre.org/`. 47

[53] Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 3. Common Criteria Development Board (2009). Available at `http://www.commoncriteriaportal.org` 49

[54] Common Methodology for Information Technology Security Evaluation, Version 3.1, Revision 3. Common Criteria Development Board (2009) 49, 214

[55] Common Vulnerabilities and Exposures (CVE) - The Standard for Information Security Vulnerability Names. MITRE Corporation. Available at `http://cve.mitre.org/`. 46, 47

[56] Common Weakness Enumeration (CWE) - A Community Developed Dictionary of Software Weakness Types. MITRE Corporation. MITRE Corporation. Available at `http://cwe.mitre.org`. 47

[57] D. Cooper, S. Santesson, S. Farrel, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure. Certificate and Certificate Revocation List (CRL) Profile. Internet Engineering Task Force – Request for Comments 5280 (2008) 22, 24, 92, 117, 141, 285

[58] D. Coppersmith. Another Birthday Attack. Advances in Cryptology - Proceedings of Crypto '85, LNCS 218, pp. 369–378, Springer–Verlag (1985) 271

[59] J. C. Cruellas, G. Karlinger, D. Pinkas, and J. Ross. XML Advanced Electronic Signatures (XAdES). World Wide Web Consortium (W3C) (2003) 24, 27, 138

[60] D. Cruz Rivero. Eficacia formal y probatoria de la firma electrónica. Marcial Pons (Ed.). ISBN: 84–9768–353–6 (2006) 6, 28, 30, 31

[61] P. Dasgupta, K. Chatha, and S. K. S. Gupta. Vulnerabilities of PKI based Smartcards. IEEE Military Communications Conference (MILCOM 2007), pp. 1–5 (2007) 56, 261

[62] N. Davis. Secure Software Development Life Cycle Processes: A Technology Scouting Report. Technical Note CMU/SEI-2005-TN-024 (2005) 53

[63] W. Diffie and M. Hellman. New Directions in Cryptography. IEEE Transactions of Information Theory, 22(6), pp. 644–654 (1976) 4, 21

[64] X. Ding, D. Mazzocchi, and G. Tsudik. Experimenting with Server-Aided Signatures. Network and Distributed Systems Security Symposium (NDSS '02) (2002) 62

[65] Document Object Model. W3C. Available at `http://www.w3.org/DOM/` 293

[66] D. Dolev and A. Yao. On the Security of Public-Key Protocols. IEEE Transactions on Information Theory, 2(29) (1983) 199

[67] P. H. Drielsma, S. Mödersheim. The ASW Protocol Revisited: A Unified View. Electronic Notes in Theoretical Computer Science (ENTCS), 125(1), pp. 145–161 (2005) 216

[68] Electronic Signatures in Global and National Commerce Act (e-sign). Federal Trade Commission, Department of Commerce. United States of America. June 30, 2000. Available at `www.senate.gov/search/index.htmlunderS.761inthe106thCongress`. 28

[69] EMV Integrated Circuit Card Specification for Payment Systems. Book 3: Application Specification. Version 4.1. EMVCo (2007) 37

[70] ETSI TR 102 038 v1.1.1. TC Security – Electronic Signatures and Infrastructures (ESI); XML format for signature policies. European Telecommunications Standards Institute (2002) 26, 28, 158

[71] ETSI TR 102 041 v1.1.1 Signatures Policies Report. European Telecommunications Standards Institute (2002) 8, 11, 25, 27, 53, 116, 146, 147, 153

[72] ETSI TR 102 045 v1.1.1. Electronic Signatures and Infrastructures (ESI); Signature policy for extended business model. European Telecommunications Standards Institute (2003) 9, 28, 115, 125, 126

[73] ETSI TR 102 272 v1.1.1. Electronic Signatures and Infrastructures (ESI); ASN.1 format for signature policies. European Telecommunications Standards Institute (2003) 26, 28, 117, 118, 121, 122, 124, 128, 129, 158

[74] ETSI TS 101 733 v1.7.4. Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES). European Telecommunications Standards Institute (2008) 24, 27, 83, 118, 124, 131, 138, 139, 142, 159, 288

[75] ETSI TS 101 903 v1.3.2. XML Advanced Electronic Signatures (XAdES). European Telecommunications Standards Institute (ETSI) (2006) 24, 27, 83, 138, 139, 159, 190, 191, 192, 194, 195, 288, 292

[76] European Directive 1999/93/CE of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures (1999) 4, 28, 29, 70, 72

[77] P. Fahn and P. Pearson. IPA: A New Class of Power Attacks. 1st International Workshop on Cryptographic Hardware and Embedded Systems (CHES 1999), LNCS 1717, pp. 173–186, Springer–Verlag (1999) 56, 265

[78] P. Fites and M. P. Kratz. Information Systems Security: A Practitioner's Reference. Van Nostrand Reinhold, New York (1993) 275

[79] S. Forrest, A. Somayaji, and D. H. Ackley. Building diverse computer systems. 6th Workshop on Hot Topics in Operating Systems (1997) 177

[80] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. Cryptographic Hardware and Embedded Systems (CHES 2001), LNCS 2162, pp. 251–261, Springer–Verlag (2001) 56, 266

[81] S. Garriss, R. Sailer, R. Caceres, L. van Doorn, S. Berger, and X. Zhang. Towards Trustworthy Kiosk Computing. IEEE Workshop on Mobile Computing Systems and Applications (HotMobile 2007), pp. 41–45 (2007) 59

[82] D. Geer, R. Bace, P. Gutmann, P. Metzger, C. P. Pfleeger, J. S. Quarterman, and B. Schneier. CyberInsecurity: The Cost of Monopoly. Report, CCIA (2003) 177

[83] D. K. Gifford, L. C. Stewart, A. C. Payne, and G. W. Treese. Payment switches for open networks. 40th IEEE Computer Society International Conference (COMPCON'95), pp. 26–31 (1995) 58

[84] P. Girard and J-L. Giraud. Software attacks on smart cards. Information Security Technical Report, 8(1), pp. 55–66 (2003) 46, 56, 260, 261

[85] Y. Glouche, T. Genet, O. Heen, and O. Courtay. A Security Protocol Animator Tool for AVISPA. ARTIST2 Workshop on Security Specification and Verification of Embedded Systems (2006) 199

[86] Guías STIC del Centro Criptológico Nacional. Available at `https://www.ccn-cert.cni.es/index.php`. Note: Most are classified documents not publicly available. 52

[87] Guide for the Security Certification and Accreditation of Federal Information Systems. NIST Special Publication SP 800-37 (2004) 52

[88] P. Gutmann. BreakMS - Break Microsoft Private Key Encryption with a dictionary attack (1997). Available at `http://www.artofhacking.com/tucops/etc/crypto/live/aoh_breakms.htm`. 260

[89] P. Gutmann. How to recover private keys for Microsoft Internet Explorer, Internet Information Server, Outlook Express, and many others - or - Where do your encryption keys want to go today? (1998). Available at `http://www.cs.auckland.ac.nz/~pgut001/pubs/breakms.txt`. 260

[90] S. Hansman. A Taxonomy of Network and Computer Attack Methodologies. Technical Report, Department of Computer Science and Software Engineering, University of Canterbury, Christchurch (2003) 42

[91] S. Hansman and R. Hunt. A taxonomy of network and computer attacks. Computers & Security 24, pp. 31–43 (2005) 46

[92] M. Hartmann and L. Eckstein. TruPoSign: A trustworthy and mobile platform for electronic signatures. Information Security & Business, Vieweg Verlag (2003) 57

[93] J. L. Hernández-Ardieta, F. H. Álvarez, and C. J. Suárez. Systems and Methods for Using Cryptographic Keys. Application number: US 12/424324. Priority date: 15/04/2009 142, 210

[94] J. L. Hernández-Ardieta, A. I. González-Tablas, and B. R. Álvarez. Protocolo de intercambio justo para comercio electrónico basado en políticas de firma. Actas del II Simposio sobre Seguridad Informática SSI'07 (CEDI 2007), pp. 249–256, Ed. Thomson (2007) 211

[95] J. L. Hernandez-Ardieta, A. I. Gonzalez-Tablas, and B. R. Alvarez. An Optimistic Fair Exchange Protocol based on Signature Policies. Computers & Security, 27(7-8), pp. 309–322 (2008) 11, 119, 211

[96] J. L. Hernández-Ardieta, A. I. González-Tablas, B. R. Álvarez, and A. Ribagorda. Aumento de la Fiabilidad de Evidencias Digitales mediante la División del Entorno de Firma en un Protocolo de Intercambio Justo. V Congreso Iberoamericano de Seguridad Informática (CIBSI 2009) (2009) 210, 211

[97] J. L. Hernández-Ardieta, A. I. González-Tablas, and B. Ramos. Repudio de firmas electrónicas en Infraestructuras de Clave Pública. Actas de la X Reunión sobre Criptología y Seguridad Informática (X RECSI 2008), pp. 595–606 (2008) 210

[98] J. L. Hernandez-Ardieta, A. I. Gonzalez-Tablas, and B. Ramos. Formal Validation of OFEPSP+ with AVISPA. Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security, LNCS 5511, pp. 124–137, Springer–Verlag (2009) 12, 211

[99] J. L. Hernández-Ardieta, A. I. González-Tablas, and B. Ramos. Taxonomía de ataques a entornos de creación de firmas electrónicas. Actas de la XI Reunión sobre Criptología y Seguridad Informática (XI RECSI 2010) (2010) 210

[100] J. L. Hernandez-Ardieta, A. I. Gonzalez-Tablas, B. Ramos, and A. Ribagorda. On the Need to Divide the Signature Creation Environment. International Conference on Security and Privacy (SECRYPT 2009), pp. 375–379 (2009) 210

[101] J. L. Hernandez-Ardieta, A. I. Gonzalez-Tablas, B. Ramos, and A. Ribagorda. Extended Electronic Signature Policies. 2nd ACM International Conference on Security of Information and Networks (SIN 2009), pp. 268–277, ACM Press (2009) 210

[102] B. Hill. A Taxonomy of Attacks against XML Digital Signatures & Encryption (2004). Available at `http://www.isecpartners.com/files/iSEC_HILL_AttackingXMLSecurity_Handout.pdf` 44

[103] J. D. Howard and T. A. Longstaff. A Common Language for Computer Security Incidents. Technical Report, Sandia National Laboratories (1998) 43

[104] IAIK XML Advanced Electronic Signatures (XAdES) add-on for XML Security Toolkit (XSECT). Available at `http://jce.iaik.tugraz.at/sic/products/xml_security/xades` 190

[105] IAIK XML Security Toolkit (XSECT). Available at `http://jce.iaik.tugraz.at/sic/products/xml_security/xsect` 190

[106] V. M. Igure and R. D. Williams. Taxonomies of attacks and vulnerabilities in computer systems. IEEE Communications Surveys and Tutorials, 10(1–4), pp. 6–19 (2008) 42

[107] Information Security Committee, Section of Science & Technology, American Bar Association – Digital Signatures Guidelines (1996). Available at `http://www.abanet.org/scitech/ec/isc/dsgfree.html` 31

[108] Informe sobre nuevas tecnologías de autoprotección de clientes y sistemas. SECUWARE, Jorge López Hernández-Ardieta (Ed.), CENIT-Segur@, ref. CENIT-2007 2004 (2008) 57

[109] Institute for Applied Information Processing and Communications (IAIK). `http://jce.iaik.tugraz.at/sic/products/xml_security` 190

[110] Introducing the First Virtual Internet Payment System for Information Commerce. First Virtual, Inc. (1994) 58

[111] ISO/IEC 13335-1, Information technology – Security techniques – Management of information and communications technology security – Part 1: Concepts and models for information and communications technology security management. International Organization for Standardization (2004) 3

[112] ISO/IEC DIS 13888-1, Information technology – Security techniques – Non repudiation – Part 1: General model. International Organization for Standardization (2009) 3, 8, 33, 34, 35, 37

[113] ISO/IEC 13888-2, Information technology – Security techniques – Non-repudiation – Part 2: Mechanisms using symmetric techniques. International Organization for Standardization (1998) 4, 34

[114] ISO/IEC 13888-3, Information technology – Security techniques – Non repudiation – Part 3: Mechanisms Using Asymmetric Techniques. International Organization for Standardization (2009) 4, 34, 35, 37

[115] ISO/IEC 14516, Information technology – Security techniques – Guidelines for the use and management of Trusted Third Party services. International Organization for Standardization (2002) 35

[116] ISO/IEC 15408–1, Information technology - Security techniques - Evaluation criteria for IT security - Part 1:Introduction and general model. ISO/IEC JTC 1/SC 27 (2009) 7, 49

[117] ISO/IEC 15408–2, Information technology - Security techniques - Evaluation criteria for IT security - Part 2:Security functional components. ISO/IEC JTC 1/SC 27 (2008) 49

[118] ISO/IEC 15408–3, Information technology - Security techniques - Evaluation criteria for IT security - Part 3:Security assurance components. ISO/IEC JTC 1/SC 27 (2008) 49

[119] ISO/IEC 18045, Information technology - Security techniques - Methodology for IT security evaluation. ISO/IEC JTC 1/SC 27 (2008) 49

[120] ISO/IEC 19791, Information technology - Security techniques - Security assessment of operational systems. ISO/IEC JTC 1/SC 27/WG 3 (2009) 7, 52

[121] ISO/IEC 27005, Information technology - Security Techniques - Information security risk management. ISO/IEC JTC 1/SC 27 (2008) 52

[122] ISO/IEC 8824–1, Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation. International Organization for Standardization (2002) 24

[123] ISO 7498-2, Information processing system – Open systems interconnection – Basic reference model – Part 2: Security architecture. International Organization for Standardization (1989) 3, 21, 69

[124] ITU–T Recommendation X.690. Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER). ITU–T (2002) 117

[125] W. Jansen. Directions in Security Metrics Research. NISTIR 7564, National Institute of Standards and Technology (2009) 213, 214

[126] Java Security. Available at `http://java.sun.com/j2se/1.5.0/docs/guide/security/index.html` 190

[127] M. Joye, P. Paillier, and B. Schoenmarkers. On Second-Order Differential Power Analysis. 7th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2005), LNCS 3659, Springer–Verlag, pp. 293-308 (2005) 264

[128] JUnit testing framework. Available at `http://www.junit.org/` 188

[129] A. Jøsang and B. AlFayyadh. Robust WYSIWYS: A method for ensuring that What You See Is What You Sign. 6th Australasian conference on Information security (AISC 2008), pp. 53-58 (2008) 60, 61

[130] A. Jøsang, D. Povey, and A. Ho. What You See is Not Always What You Sign. Australian UNIX User Group (2002) 51, 251, 252, 275, 276, 279

[131] K. Kain. Electronic Documents and Digital Signatures. Doctoral Thesis (2003) 44, 51, 253, 254, 255, 280, 281, 282

[132] R. Kehr, J. Posegga, and H. Vogt. PCA: Jini-based Personal Card Assistant. Secure Networking - CQRE [Secure]'99, LNCS 1740, pp. 64–75. Springer–Verlag (1999) 57

[133] J. Kelsey and T. Kohno. Herding Hash Functions and the Nostradamus Attack. Advances in Cryptology - EUROCRYPT 2006, LNCS 4004, pp. 183–200 (2006) 271

[134] J. Kelsey and B. Schneier. Second preimages on n-bit hash functions for much less than 2n work. EUROCRYPT 2005, LNCS 3494, pp. 474–490 (2005) 271

[135] R. Kilian-Kehr and J. Posegga. Smart Cards in Interaction: Towards Trustworthy Digital Signatures. 5th Conference on Smart Card Research and Advanced Application Conference (CARDIS 2002), 5, pp. 11–18 (2002) 56, 62

[136] F. Klay and L. Vigneron. Automatic Methods for Analyzing Non-repudiation Protocols with an Active Intruder. 5th International Workshop on Formal Aspects in Security and Trust (FAST 2008), LNCS 5491, pp. 192–209, Springer (2009) 216

[137] V. Klima. Tunnels in hash functions: MD5 collisions within a minute. IACR ePrint archive (2006) 271

[138] P. C. Kocher. Timing attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. Advances in Cryptology - CRYPTO '96, LNCS 1109, pp. 104–113, Springer–Verlag (1996) 261, 262

[139] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. CRYPTO 1999, LNCS 1666, pp. 388–397, Springer–Verlag (1999) 263, 264

[140] O. Kömmerling and M. G. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. Proc. USENIX Workshop on Smartcard Technology (WOST'99), 1999. 81

[141] S. Kremer and O. Markowitch. Optimistic non-repudiable information exchange. 21st Symposium on information theory in the Benelux. Wassenaar, The Netherlands: Werkgemeenschap Informatieen Communicatietheorie, Enschede, pp. 139–146 (2000) 39

[142] S. Kremer, O. Markowitch, and J. Zhou. An intensive survey of fair non-repudiation protocols. Computer Communications, 25, pp. 1601–1621 (2002) 39, 119, 146

[143] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. S. Choi. A taxonomy of computer program security flaws. ACM Computing Surveys, 26(3), pp. 211–254 (1994) 42

[144] H. Langweg and T. Kristiansen. Extending the Trusted Path in Client-Server Interaction. CoRR abs/cs/0611102 (2006) 64

[145] H. Langweg. Malware attacks on electronic signatures revisited. Sicherheit 2006: Jahrestagung Fachbereich Sicherheit der Gesellschaft für Informatik, LNI, pp. 244–255 (2006) 248, 249, 250, 251, 276, 277, 278

[146] H. Langweg and E. Snekkenes. A Classification of Malicious Software Attacks. 23rd IEEE International Performance, Computing, and Communications Conference (2004) 42

[147] T-H. Le, C. Canovas, and J. Clediere. An overview of side channel analysis attacks. ACM Symposium on Information, Computer and Communications Security (ASIACCS 2008), pp. 33–43 (2008) 56, 263, 264, 265

[148] T.H. Le, J. Clédière, C. Canovas, C.Servière, J.L. Lacoume, and B. Robisson. A proposition for Correlation Power Analysis enhancement. 8th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2006), LNCS 4249, pp. 174–186, Springer–Verlag (2006) 264

[149] B. Lee and K. Kim. Fair Exchange of Digital Signatures using Conditional Signature. Symposium on Cryptography and Information Security (SCIS 2002) (2002) 38, 55

[150] S. Lefranc and D. Naccache. Cut and paste attacks with java. Cryptology ePrint Archive, Report (2002). Available at `http://eprint.iacr.org/`. 244, 288

[151] D. Lekkas. Establishing and managing trust within the Public Key Infrastructure. Computer Communications, 26(16), pp. 1815–1825 (2003) 210

[152] D. Lekkas, S. Gritzalis, and L. Mitrou. Withdrawing a declaration of will: Towards a framework for Digital Signature Revocation. Internet Research, 15(4) pp. 400–420 (2005) 55

[153] A. K. Lenstra and A. Shamir. Analysis and optimization of the TWINKLE factoring Device. EUROCRYPT 2000, LNCS 1807, pp. 35–52, Springer–Verlag (2000) 273

[154] Ley Española 59/2003, de 19 de Diciembre, de Firma Electrónica (2003) 4, 28, 29, 31

[155] U. Lindqvist and E. Johsson. How to Systematically Classify Computer Security Intrusions. IEEE Security and Privacy, pp. 154–163 (1997) 42

[156] H. Lisha, N. Zhang, L. He, and I. Rogers. Secure M-commerce Transactions: A Third Party Based Signature Protocol. 3rd International Symposium on Information Assurance and Security, pp. 3–8 (2007) 62

[157] P. A. Loscocco, S. D. Smalley, P. A. Muckelbauer, R. C. Taylor, S. J. Turner, and J. F. Farrell. The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments. 21st National Information Systems Security Conference, pp. 303–314 (1998) 52

[158] D. L. Lough. A taxonomy of computer attacks with applications to wireless networks. PhD thesis, Virginia Polytechnic Institute and State University (2001) 43, 166

[159] J. Loughry and D. A. Umphress. Information Leakage from Optical Emanations. ACM Transactions on Information and System Security, 5(3), pp. 262–289 (2002) 275

[160] Malicious Software (malware): a Security Threat to the Internet Economy. Organization for Economic Co-operation and Development (OECD) (2008) 7, 81

[161] S. Matamoros, J. Martinez, and A. Maña. Entorno de firma confiable basado en PDA. I Simposio Español de Negocio Electrónico (2001) 57

[162] A. Matthews. Low cost attacks on smart cards: The electromagnetic side-channel. Next Generation Security Software (2006). Available at `http://www.ngssoftware.com/research/papers/EMA.pdf`. 266

[163] T. Matsumoto. Human-computer cryptography: an attempt. 3rd ACM conference on Computer and communications security, pp. 68–75 (1996) 63

[164] J. Marchesini, S.W. Smith, and M. Zhao. Keyjacking: the surprising insecurity of client-side SSL. Computers & Security, 24(2), pp. 109–123 (2005) 255, 256, 257, 258, 259, 260

[165] U. Maurer. Intrinsic limitations of digital signatures and how to cope with them. 6th Information Security Conference (ISC'03), LNCS 2851, pp. 180–192 (2003) 54, 63

[166] M-E. Maurer and A. De Luca. SeCuUI: Autocomplete your Terminal Input. International Conference on Human-Computer Interaction with Mobile Devices and Services (2009) 59

[167] A. McCullagh and W. Caelli. Non-repudiation in the digital Environment. First Monday, 5(8) (2000). Available at `http://firstmonday.org/issues/issue5_8/mccullagh/index.html`. 30

[168] J. M. McCune, A. Perrig, and M. K. Reiter. Bump in the ether: a framework for securing sensitive user input. USENIX Annual Technical Conference, pp. 185–198 (2006) 60

[169] E. Mills. Secure Software? Experts Say it's no Longer a Pipe Dream. CNET News, April 20 (2009) 53

[170] Mobile Security Report. McAfee (2009) 57

[171] C. van den Munckhof. Fooling the Trusted Platform Module using a simple reset. Seminar at the Technische Universiteit Eindhoven (2009) 62

[172] T. Murmann and H. Rossnagel. How Secure Are Current Mobile Operating Systems? 8th IFIP/CMS 2004, IFIP 175, pp. 47–58 (2004) 57

[173] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. Internet X.509 Public Key Infrastructure. Online Certificate Status Protocol – OCSP. Internet Engineering Task Force – Request for Comments 2560 (1999) 23, 91, 92

[174] J. Müller-Quade and S. Röhrich. What you see is what you sign. Heidelberger Innovation Forum (2007) 60, 61

[175] A. M. Nadal and J. L. F. Gomila. Delimitación de Responsabilidades en Caso de Revocación de un Certificado de Firma Electrónica. I Simposio Español de Negocio Electrónico (2001) 31

[176] M. Naor and B. Pinkas. Visual Authentication and Identification. 17th Annual International Cryptology Conference on Advances in Cryptology, LNCS 1294, pp. 322–336 (1997) 63

[177] National Vulnerability Database (NVD). National Institute of Standards and Technology (NIST). Available at `http://nvd.nist.gov/home.cfm`. 46

[178] F. Nentwich, E. Kirda, and C. Kruegel. Practical Security Aspects of Digital Signature Systems. International Secure Systems Lab, TR-Seclab-0606-001 (2006) 247

[179] Y. Okada, Y. Manabe, and T. Okamoto. An optimistic fair exchange protocol and its security in the universal composability framework. International Journal of Applied Cryptography, 1(1), pp. 70–77 (2008) 39

[180] A. Oprea, D. Balfanz, G. Durfee, and D. K. Smetters. Securing a remote terminal application with a mobile trusted device. Computer Security Applications Conference, pp. 438–447 (2004) 59

[181] H. Pagnia and F. C. Gärtner. On the impossibility of fair exchange without a Trusted Third Party. Technical Report TUD-BS-1999-02, Darmstadt University of Technology, Department of Computer Science, Darmstadt, Germany (1999) 39

[182] S. C. Payne. A Guide to Security Metrics. SANS Security Essentials GSEC Practical Assignment, Version 1.2e (2006) 213

[183] A. Pellegrini, V. Bertacco, and T. Austin. Fault-Based Attack of RSA Authentication. Design, Automation and Test in Europe Conference (DATE-2010) (2010) 267

[184] Personal Information Protection and Electronic Documents Act. Department of Justice. Government of Canada. May 30th, 2008 28, 29, 30

[185] Phishing Activity Trends Report. APWG (2009) 7, 81

[186] F. Piessens. A taxonomy of causes of software vulnerabilities in Internet software. 13th International Symposium on Software Reliability Engineering, pp. 47–52 (2002) 42

[187] D. Pinkas, N. Pope, and J. Ross. CMS Advanced Electronic Signatures (CAdES). Internet Engineering Task Force – Request for Comments 5126 (2008) 24, 118, 124, 131, 138, 139, 142

[188] F. Piva. Timeliness in fair exchange protocols. Workshop on Formal Methods in Cryptography (2009) 38

[189] PKCS #11 v2.30: Cryptographic Token Interface Standard. RSA Laboratories (2009) 51

[190] PPSCVA-T1, EAL1. Perfil de Proteccion para la aplicación de creación y verificación de firma electrónica Tipo 1, con control exclusivo de los interfaces con el firmante y con el nivel de evaluación de los requisitos de seguridad EAL1. Instituto Nacional de Tecnologías de la Comunicación (INTECO) (2008) 50

[191] PPSCVA-T1, EAL3. Perfil de Proteccion para la aplicación de creación y verificación de firma electrónica Tipo 1, con control exclusivo de los interfaces con el firmante y con el nivel de evaluación de los requisitos de seguridad EAL3. Instituto Nacional de Tecnologías de la Comunicación (INTECO) (2008) 50

[192] PPSCVA-T2, EAL1. Perfil de Proteccion para la aplicación de creación y verificación de firma electrónica Tipo 2, con nivel de evaluación de los requisitos de seguridad EAL1. Instituto Nacional de Tecnologías de la Comunicación (INTECO) (2008) 50

[193] PPSCVA-T2, EAL3. Perfil de Proteccion para la aplicación de creación y verificación de firma electrónica Tipo 2, con nivel de evaluación de los requisitos de seguridad EAL3. Instituto Nacional de Tecnologías de la Comunicación (INTECO) (2008) 50

[194] Provisional Signature Schemes. United States Patent Application 12/389295 (2009) 62

[195] J.-J. Quisquater and D. Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-measures for Smart Cards. International Conference on Research in Smart Cards (E-smart 2001), LNCS 2140, pp. 200–210, Springer–Verlag (2001) 266

[196] A. J. Rae and L. P. Wildman. A Taxonomy of attacks on secure devices. 4th Australian Information Warfare and IT Security Conference, pp. 251–264 (2003) 42, 45

[197] I. Ray and I. Ray. Fair exchange in e-commerce. ACM SIGecom Exchange, 3(2), pp. 9–17 (2002) 11, 38, 119

[198] Real-World Malware Statistics. SurfRight (2009) 7, 81

[199] R. L. Rivest. Issues in Cryptography. Computers, Freedom, and Privacy Conference (2001) 54

[200] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), pp. 120–126 (1978) 21

[201] M. Roe. Cryptography and Evidence. Doctoral Thesis. University of Cambridge Computer Laboratory (1997) 54

[202] S. J. Ross, J. L. Hill, M. Y. Chen, A. D. Joseph, D. E. Culler, and E. A. Brewer. A Composable Framework for Secure Multi-Modal Access to Internet Services from Post-PC Devices. Mobile Networks and Applications 7, pp. 389–406, Kluwer Academic Publishers (2002) 59

[203] J. Ross, D. Pinkas, and N. Pope. Electronic Signature Policies. Internet Engineering Task Force – Request For Comments 3125 (Experimental) (2001) 11, 25, 27, 116, 117, 118, 121, 122, 124, 128, 129

[204] A. Ruiz-Martínez, D. Sánchez-Martínez, M. Martínez-Montesinos, and A. F. Gómez-Skarmeta. A Survey of Electronic Signature Solutions in Mobile Devices. Journal of Theoretical and Applied Electronic Commerce Research, 2(3), pp. 94–109 (2007) 57

[205] J. Rushby. Critical system properties: Survey and taxonomy. Technical Report CSL-93-01, Computer Science Laboratory. SRI International (1994) 81

[206] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and implementation of a TCG-based integrity measurement architecture. USENIX Security Symposium (2004) 59

[207] K. Scheibelhofer. What You See Is What You Sign – Trustworthy Display of XML Documents for Signing and Verification. IFIP TC6/TC11 International Conference on Communications and Multimedia Security Issues of the New Century, pp. 35 (2001) 51

[208] W. Schindler. A Timing Attack against RSA with the Chinese Remainder Theorem. Cryptographic Hardware and Embedded Systems (CHES 2000), LNCS 1965, pp. 110–125. Springer–Verlag (2000) 56, 262

[209] W. Schindler, K. Lemke, and C. Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. 7th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2005), LNCS 3659, pages 30–46 (2005) 265

[210] F. Schneider and K. P. Birman. The monoculture of risk put into context. IEEE Security and Privacy, 7(1), pp. 14–17 (2009) 177

[211] SET Secure Electronic Transaction (TM). Version 1.0. Book 1: Business description. Book 2: Programmer's guide. Book 3: Formal protocol definition. VISA (1997) 37

[212] A. Shamir and E. Tromer. Special-Purpose Hardware for Factoring: the NFS Sieving Step. Invited talk at the Workshop on Special Purpose Hardware for Attacking Cryptographic Systems (SHARCS) (2005) 273

[213] C. E. Shannon. Communication theory of secrecy systems. Bell System Technical Journal, 28(4), pp. 656–715 (1949) 179

[214] M-H. Shao, G. Wang, and J. Zhou. Some common attacks against certified email protocols and the countermeasures. Computer Communications, 29, pp. 2759-2769 (2006) 156

[215] R. Sharp, A. Madhavapeddy, R. Want, and T. Pering. Enhancing Web Browsing Security on Public Terminals Using Mobile Composition. International Conference on Mobile Systems, Applications And Services, pp. 94–105 (2008) 59

[216] R. Sharp, J. Scott, and A. Beresford. Secure mobile computing via public terminals. Pervasive 2006, pp. 238–253. Springer–Verlag (2006) 59

[217] D-H. Shih, B. Lin, H-S. Chiang, and M-H. Shih. Security aspects of mobile phone virus: A critical survey. Industrial Management & Data Systems, 108(4), pp. 478–494 (2008) 57

[218] R. Shirey. Internet Security Glossary, Version 2. Internet Engineering Task Force – Request for Comments 4949 (2007) 3, 68, 80

[219] A. Spalka, A. B. Cremers, and H. Langweg. The Fairy Tale of What You See Is What You Sign. Trojan Horse Attacks on Software for Digital Signatures. IFIP Working Conference on Security and Control of IT in Society-II, pp. 75–86 (2001) 98

[220] A. Spalka, A. B. Cremers, and H. Langweg. Protecting the Creation of Digital Signatures with Trusted Computing Platform Technology Against Attacks by Trojan Horse Programs. IFIP TC11 16th International Conference on Information Security (2001) 61, 62

[221] A. Spalka, A. B. Cremers, and H. Langweg. Trojan Horse Attacks on Software for Electronic Signatures. Informatica 26, pp. 191–204 (2002) 245, 246

[222] SSE-CMM: Systems Security Engineering Capability Maturity Model. International Systems Security Engineering Association (ISSEA) (2008) 213

[223] H. Tanaka. Evaluation of Information Leakage via Electromagnetic Emanation and Effectiveness of Tempest. IEICE Transactions on Information and Systems, 91(5), pp. 1439–1446 (2008) 56, 266

[224] The side-channel cryptanalysis lounge. European Network of Excellence in Cryptology. Available at `http://www.crypto.ruhr-uni-bochum.de/en_sclounge.html`. 45

[225] K. Tiri. Side-channel attack pitfalls. 44th Annual Conference on Design Automation, pp. 15–20 (2007) 263

[226] Trusted Computing Group. TPM Main Part 1 Design Principles, Specification Version 1.2 Revision 94 (2006) 59, 112

[227] Trusted Computing Group. TCG Mobile Trusted Module Specification, Version 1.0 Revision 1 (2007) 112

[228] M. Turuani. The CL-Atse Protocol Analyser. 17th International Conference on Rewriting Techniques and Applications, LNCS 4098, pp. 277–286, Springer (2006) 199

[229] UNCITRAL Model Law on Electronic Signatures with Guide to Enactment, United Nations (2001) 4, 28, 29, 30, 31

[230] G. Wang and S. Wang. Preimage Attack on Hash Function RIPEMD. 5th International Conference on Information Security Practice and Experience, LNCS 5451, pp. 274–284 (2009) 272

[231] X. Wang and H. Yu. How to Break MD5 and Other Hash Functions. Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, LNCS 3494, pp. 19–35 (2005) 270

[232] W3C Recommendation – Extensible Markup Language (XML) 1.0. Fourth Edition (2006) 24

[233] W3C Recommendation – XML Signature Syntax and Processing. Second Edition (2008) 194, 292

[234] S. Yang, S. Y. W. Su, and H. Lam. A non-repudiation message transfer protocol for collaborative e-commerce. International Journal of Business Process Integration and Management, 1(1) (2005) 39

[235] Y-S. Yeh, T-Y. Huang, H-Y. Lin, and Y-H. Chang. A Study on Parallel RSA Factorization. Journal of Computers, 4(2), pp. 112–118 (2009) 273

[236] H. Zhang, M. Kudo, K. Matsuura, and H. Imai. A model for signature revocation. International Symposium on Information Theory and Its Applications (ISITA 2002), Xi'an, PRC, pp. 455–458 (2002) 55

[237] J. Zhou and D. Gollmann. A Fair Non-repudiation Protocol. IEEE Symposium on Research in Security and Privacy, pp. 55–61 (1996) 11, 39

[238] J. Zhou and D. Gollmann. Evidence and Non-repudiation. Journal of Network and Computer Applications, 20(3), pp. 267–281 (1997) 4, 34, 35

[239] L. Zhuang, F. Zhou, and J. D. Tygar. Keyboard Acoustic Emanations Revisited. ACM Conference on Computer and Communications Security, pp. 373–382 (2005) 269

[240] F. Zumbiehl. Collisions in PDF Signatures (2010). Available at `http://pdfsig-collision.florz.de/` 278

Appendix A

Publications and Patents

The following list contains the references to the papers that resulted from this thesis and that have been published in scientific journals and conferences. The publications are listed following a chronological order:

1. Jorge L. Hernández-Ardieta, Ana Isabel González-Tablas, y Benjamín Ramos Álvarez. *Protocolo de intercambio justo para comercio electrónico basado en políticas de firma.* Actas del II Simposio sobre Seguridad Informática SSI'07 (CEDI 2007), pp. 249–256, Editorial Thomson. Zaragoza (2007)

2. J. L. Hernández-Ardieta, A. I. González-Tablas, y B. Ramos. *Repudio de firmas electrónicas en Infraestructuras de Clave Pública.* Actas de la X Reunión sobre Criptología y Seguridad Informática (X RECSI 2008), pp. 595–606. Salamanca (2008)

3. Jorge L. Hernandez-Ardieta, Ana I. Gonzalez-Tablas, and Benjamin Ramos. *An Optimistic Fair Exchange Protocol based on Signature Policies.* Computers & Security 27(7-8), pp. 309–322. Elsevier (Eds.) (2008) [JCR 2008:1,028]

4. Jorge L. Hernandez-Ardieta, Ana I. Gonzalez-Tablas, and Benjamin Ramos. *Formal Validation of OFEPSP+ with AVISPA.* Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security, LNCS 5511, pp. 124–137. Springer-Verlag (2009)

5. Jorge L. Hernandez-Ardieta, Ana I. Gonzalez-Tablas, Benjamin Ramos, and Arturo Ribagorda. *On the Need to Divide the Signature Creation Environment.* International Conference on Security and Privacy (SECRYPT 2009), pp. 375–379. Milan (2009)

6. Jorge L. Hernandez-Ardieta, Ana I. Gonzalez-Tablas, Benjamin Ramos, and Arturo Ribagorda. *Extended Electronic Signature Policies.* 2nd ACM International

Conference on Security of Information and Networks (SIN 2009), pp. 268–277, ACM Press. North Cyprus (2009)

7. Jorge L. Hernández-Ardieta, Ana Isabel González-Tablas, Benjamín Ramos Álvarez, y Arturo Ribagorda. *Aumento de la Fiabilidad de Evidencias Digitales mediante la División del Entorno de Firma en un Protocolo de Intercambio Justo.* V Congreso Iberoamericano de Seguridad Informática (CIBSI 2009). Montevideo (2009)

8. Jorge López Hernández-Ardieta, Ana Isabel González-Tablas, y Benjamín Ramos. *Taxonomía de ataques a entornos de creación de firmas electrónicas.* Actas de la XI Reunión sobre Criptología y Seguridad Informática (XI RECSI 2010), pp. 85–90. Tarragona (2010)

Finally, the patent application which results were generated in the framework of the current thesis is quoted.

1. Jorge López Hernández-Ardieta, Fernando Hernández Álvarez, and Carlos Jiménez Suárez. *Systems and Methods for Using Cryptographic Keys.* Application number: US 12/424324. Priority date: 15/04/2009

Appendix B

Classified Attacks on Digital Signatures

This Appendix includes 112 attacks on digital signatures that have been classified using the taxonomy and method of classification proposed in Chapter 6.

The surveyed attacks are a mix of real-world attacks on specific commercial products and devised theoretical attacks that could be put into practice. Some of the 31 attacks considered on the verification stage are firstly proposed here.

A representative name, the reference to the attack, a short description and possible countermeasures are provided for each classified attack. It is important to remark that, when a countermeasure is provided, it has been considered the information given by the author of the attack and the specific conditions of the attack itself. Therefore, it should not be concluded that the countermeasure is of general applicability, and thus it should be evaluated on a case-by-case basis.

Name:	Dali attack
Source:	The Dali Attack on Digital Signature [43]
Description:	Attack based on the capability of a file of having a static polymorphic behavior. The attacker modifies the document to be signed to include a secondary content different than the purported one. Thanks to certain formats tagging, the content shown to the verifier varies depending on the file extension, and thus the application chosen to open the file. The attack is limited to the inclusion of HTML as the malicious secondary content.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.1: Dynamic content inclusion → D2-CAT2.1.1.1: Hidden code
Target(s):	D3-CAT5: Information → D3-CAT5.1: Document D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Inclusion of the signed attribute *content-type* in the electronic signature format (e.g. CAdES, XAdES)

Name:	Enhanced Dali attack
Source:	Fortifying the Dali Attack on Digital Signature [44]
Description:	Attack that enhances the Dali Attack to permit the usage of tiff and PDF formats for the contents inserted in the document to be signed.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.1: Dynamic content inclusion → D2-CAT2.1.1.1: Hidden code
Target(s):	D3-CAT5: Information → D3-CAT5.1: Document D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Use of PDF/A formats. Use of PDF Advanced Electronic Signature (PAdES) formats. Inclusion of the signed attribute *content-type* in the electronic signature format (e.g. CAdES, XAdES)

Name:	Cut and paste attack
Source:	Cut and paste attacks with Java [150]
Description:	Attack focused on using a malicious applet to modify regions of the visualization area of a web browser while surfing through the Internet. This attack could be mounted to modify a malicious document visualized by the signer before computing the signature in order to fit with the expected one.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions

Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.2: Content modification
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.1: External application → D3-CAT2.1.1.2: User level application
Countermeasures:	Disable Java Virtual Machine

Name:	PIN retrieval
Source:	Trojan Horse Attacks on Software for Electronic Signatures [221]
Description:	The attack is carried out on two of the most deployed signature software in Germany. The attacker obtains a handle to the PIN edit control in a Windows operating system environment. Once the user has entered the PIN, the attacker is able to retrieve it and start as many signing processes as desired. The authors provide an example in Delphi source code.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.1: Compromise of the signer authentication data (SAD) → D2-CAT4.1.2: SAD interception → D2-CAT4.1.2.3: Endpoint compromise
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA
Countermeasures:	Avoid handles belonging to applications different than the one that created the PIN window. Use of specialized hardware (e.g. keyboard with an integrated smart card reader)

Name:	PIN retrieval in email signing software
Source:	Trojan Horse Attacks on Software for Electronic Signatures [221]
Description:	The same attack as in *PIN retrieval* is carried out on two products that consist of a signature plug-in integrated in a widely used email client software. The attacker is able to capture the PIN or password entered by the user.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.1: Compromise of the signer authentication data (SAD) → D2-CAT4.1.2: SAD interception → D2-CAT4.1.2.3: Endpoint compromise
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA
Countermeasures:	Use of a smart card with specialized hardware (e.g. keyboard with an integrated smart card reader)

Name:	PIN retrieval (with keypad)
Source:	Trojan Horse Attacks on Software for Electronic Signatures [221]

Description:	In this case, the *PIN retrieval* attack is performed on a commercial off-the-shelf product that implements a keypad for the secure input of the PIN. The attacker is able to access the permutation information, and thus is able to retrieve the PIN selected by the user.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.1: Compromise of the signer authentication data (SAD) → D2-CAT4.1.2: SAD interception → D2-CAT4.1.2.3: Endpoint compromise
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA
Countermeasures:	Use of a smart card with specialized hardware (e.g. keyboard with an integrated smart card reader)

Name:	Modification of the secure viewer's presentation
Source:	Trojan Horse Attacks on Software for Electronic Signatures [221]
Description:	This attack violates the What-You-See-Is-What-You-Sign (WYSIWYS) principle. The attack consists in manipulating the information shown by the secure viewer of a commercial off-the-shelf signature software. As a result, the attacker is able to modify the data to be signed while deceiving the user during the last confirmation step. The authors provide an example in Delphi source code.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.3: DTBS modification
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA
Countermeasures:	-

Name:	Modification of the DTBSR
Source:	Trojan Horse Attacks on Software for Electronic Signatures [221]
Description:	The attacker basically monitors the communication between the signature software and the smart card in order to modify the hash of the data sent to the card (the DTBSR).
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.4: DTBSR modification
Target(s):	D3-CAT2: Software → D3-CAT2.2: Driver → D3-CAT2.2.3: SSCDev driver
Countermeasures:	Implement a communication protocol compliant with ISO 7816 security measures

Name:	Mail forgery

Source:	Practical Security Aspects of Digital Signature Systems [178]
Description:	The attacker aims at replacing the content of an email with arbitrary data, retaining the validity of the signature. For this purpose, the attacker launches an SMTP proxy on the compromised computer to intercept the communication between the mail client and the mail server in order to change the mail content. In order to perform the Man-In-The-Middle (MITM) attack, the attacker changes the preference settings of the mail client (Thunderbird in this case) such that the connection to the mail server is redirected to the proxy. The attacker also uses a specific tool called detours for binary interception of Win32 functions, and which permit him to load a dynamic link library (DLL) with the email client. This DLL is then used to intercept the function that initiates the digital signature process, replacing the content being signed with the malicious document.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.2: Content modification
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA
Countermeasures:	-

Name:	Secure Viewer manipulation (1)
Source:	Practical Security Aspects of Digital Signature Systems [178]
Description:	In this attack, the secure viewer component delivered for use of the Austrian citizen card (trustview component of trustdesk basic suite) is compromised. The attack consists of two steps. In the first step, detours tool (see previous attack) is used to modify the Windows file access routines in the Windows runtime library in a manner that trustview shows a file different than the one for which the signature request has been made. In the second step, the attacker alters the functions that display the content to be signed in order to show the original one, that is, the one intended by the user. For this purpose, the attacker obtains a window handle to the HTML control used by trustview, being able to edit the content shown therein.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.2: Content modification
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA
Countermeasures:	-

Name:	Secure Viewer manipulation (2)
Source:	Practical Security Aspects of Digital Signature Systems [178]

Description:	In this case, the same attack as in Secure Viewer manipulation (1) is carried out on the secure viewer component of HotSign product, also delivered for use of the Austrian citizen card. Here, though no HTML control object is used, the attacker is still capable of changing the appearance of the shown document by obtaining a handler to the secure viewer window and subsequently drawing directly over the window context.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.2: Content modification
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA
Countermeasures:	-

Name:	Secure viewer compromise (1)
Source:	Malware Attacks on Electronic Signatures Revisited [145]
Description:	The attack is carried out on Deutsche Telekom T-Telesec Signet 1.6.0.4 product. The attack does not need administrator privileges and relies on design flaws, not implementation ones. In particular, by using Windows messages the Signet software can be made to display a different information regarding the data to be signed. By placing an inactive window at the top of the z-order with a fake button representing the execution of the secure viewer, the user can be tricked into clicking on it, allowing the malware to show the purported document to be signed while sending a different one to the SCD. In addition, it is possible to draw on the viewer's presentation surface, allowing an attacker that has modified the document to be signed to represent it in its original form.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.2: Content modification
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA
Countermeasures:	Fix data before it becomes obvious to an attacker that the data is relevant for signing

Name:	PCS/SC card reader communication potential compromise (1)
Source:	Malware Attacks on Electronic Signatures Revisited [145]
Description:	The attack is carried out on Deutsche Telekom T-Telesec Signet 1.6.0.4 product. The attack does not need administrator privileges and relies on design flaws, not implementation ones. In particular, the attacker installs a modified WINSCARD.DLL in the Signet's folder.

	This file is thus loaded and executed by Signet, giving access to its address space and permitting arbitrary malicious actions, like DTBSR modification.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.4: DTBSR modification
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA
Countermeasures:	Make the signing software to verify the signed code of every module used

Name:	Secure viewer compromise (2)
Source:	Malware Attacks on Electronic Signatures Revisited [145]
Description:	The attack is carried out on IT Solution trustDesk standard 1.2.0 product. The attack does not need administrator privileges and relies on design flaws, not implementation ones. In particular, the attack draws on the secure viewer's presentation to deceive the user respecting the data to be signed, while the information to be sent for signing differs.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.2: Content modification
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA
Countermeasures:	Fix data before it becomes obvious to an attacker that the data is relevant for signing

Name:	PCS/SC card reader communication potential compromise (2)
Source:	Malware Attacks on Electronic Signatures Revisited [145]
Description:	The attack is carried out on IT Solution trustDesk standard 1.2.0 product. The attack does not need administrator privileges and relies on design flaws, not implementation ones. In particular, the attacker installs a modified driver library file. Upon accessing the card reader trustDesk loads and executes the modified device driver, giving access to the software's address space and permitting arbitrary malicious actions, like DTBSR modification.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.4: DTBSR modification
Target(s):	D3-CAT2: Software → D3-CAT2.2: Driver → D3-CAT2.2.3: SSCDev driver
Countermeasures:	Make the signing software to verify the signed code of every module used

Name:	Secure viewer compromise (3)
Source:	Malware Attacks on Electronic Signatures Revisited [145]
Description:	The attack is carried out on D-Sign matrix/digiSeal 3.0.1 product. The attack does not need administrator privileges and relies on design flaws, not implementation ones. In particular, the attack modifies the viewer's presentation surface without detection to deceive the user respecting the data to be signed, while the information to be sent for signing differs.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.2: Content modification
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA
Countermeasures:	Fix data before it becomes obvious to an attacker that the data is relevant for signing

Name:	PCS/SC card reader communication potential compromise (3)
Source:	Malware Attacks on Electronic Signatures Revisited [145]
Description:	The attack is carried out on D-Sign matrix/digiSeal 3.0.1 product. The attack does not need administrator privileges and relies on design flaws, not implementation ones. As secure PIN entry is not the default option, the attacker can change the reader configuration and specify a new card terminal driver. Thereby, it is possible to load arbitrary malicious code in the software's address space, and perform the execution of malicious actions, like DTBSR modification.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.4: DTBSR modification
Target(s):	D3-CAT2: Software → D3-CAT2.2: Driver → D3-CAT2.2.3: SSCDev driver
Countermeasures:	Make the signing software to verify the signed code of every module used

Name:	Manipulated presentation of data to be signed
Source:	Malware Attacks on Electronic Signatures Revisited [145]
Description:	The attack is carried out on Ventasoft venta-sign 2.0.0.968 product. The attack does not need administrator privileges and relies on design flaws, not implementation ones. The product does not provide a secure viewer. The attack draws on the application's presentation surface, showing the user a different file name and file information, while the information to be sent for signing differs.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions

Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.2: Content modification
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA
Countermeasures:	Fix data before it becomes obvious to an attacker that the data is relevant for signing

Name:	Secure viewer compromise (4)
Source:	Malware Attacks on Electronic Signatures Revisited [145]
Description:	The attack is carried out on 2B Secure FILE 1.0 product. The attack does not need administrator privileges and relies on design flaws, not implementation ones. In particular, the attack modifies the secure viewer's presentation surface without detection to deceive the user respecting the data to be signed (very similar to Secure viewer compromise (1) attack).
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.2: Content modification
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA
Countermeasures:	Fix data before it becomes obvious to an attacker that the data is relevant for signing

Name:	Secure viewer compromise (5)
Source:	Malware Attacks on Electronic Signatures Revisited [145]
Description:	The attack is carried out on Ultimaco SafeGuard Sign & Crypt for Office 3.4.1 product. The attack does not need administrator privileges and relies on design flaws, not implementation ones. In particular, the attack modifies the secure viewer's presentation surface without detection to deceive the user respecting the data to be signed, while the information to be sent for signing differs.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.2: Content modification
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA
Countermeasures:	Fix data before it becomes obvious to an attacker that the data is relevant for signing

Name:	False positives in XML
Source:	What You See is Not Always What You Sign [130]

Description:	The attack consists in modifying external parts of the signed XML document (e.g. a referenced schema or DTD). In particular, the attack shown modifies the ATTLIST of the DTD. While the syntactic form remains the same, the semantic varies.
Goal:	D1-CAT3: Replace signed information
Method:	D2-CAT3: Modification post signature computation → D2-CAT3.1: External content
Target(s):	D3-CAT5: Information → D3-CAT5.1: Document
Countermeasures:	Application of canonicalization algorithms. Addition of all involved content, including referenced external content, in the DTBS

Name:	Font type manipulation - Fonts substitution
Source:	What You See is Not Always What You Sign [130]
Description:	An attacker can make a document have a different representation (semantic) by applying customized font types. If these font types are explicitly designed by the attacker for the document processor of the signer, and thus are not available during the verification stage, the glyph of certain characters can vary, changing the meaning of the document while maintaining the integrity of the signature. Though the authors presented this attack from the viewpoint of deceiving the verifier, this attack can be applied to deceive the signer, as described herein.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT1: Environment manipulation
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Use of formats (e.g. PDF) that include the fonts definitions inside the content of the document

Name:	Inconsistent handling of HTML table tags
Source:	What You See is Not Always What You Sign [130]
Description:	Web browsers interpret HTML and Javascript code in a different manner. Consequently, the same HTML code can be shown in different ways depending on the web browser used.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.1: Dynamic content inclusion → D2-CAT2.1.1.2: Active code
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.1: External application → D3-CAT2.1.1.2: User level application
Countermeasures:	Avoid the inclusion of dynamic content in the document to be signed

Name:	Substitution of Office document by external content using macros
Source:	Electronic Documents and Digital Signatures [131]
Description:	When opening the signed document, some active code (e.g. a macro programmed in Visual Basic for Applications for a Word document or an Excel spreadsheet) included in it substitutes the content of the document by an external content controlled by the attacker. This attack is feasible on Microsoft Office formats. As the signature is verified against the initial object, the signature integrity is not corrupted.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.1: Dynamic content inclusion → D2-CAT2.1.1.2: Active code
Target(s):	D3-CAT5: Information → D3-CAT5.1: Document
	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Avoid the inclusion of dynamic content in the document to be signed

Name:	Substitution of Office document by external content referenced by links
Source:	Electronic Documents and Digital Signatures [131]
Description:	Office documents allow users to insert material from remote documents by reference. As a result, the document only manages a link to an external object, which is loaded on demand. This characteristic permits an attacker to manipulate the linked data without corrupting the signature integrity.
Goal:	D1-CAT3: Replace signed information
Method:	D2-CAT3: Modification post signature computation → D2-CAT3.1: External content
Target(s):	D3-CAT5: Information → D3-CAT5.1: Document
	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Avoid the inclusion of links to external content in the document to be signed

Name:	External queries in Excel
Source:	Electronic Documents and Digital Signatures [131]
Description:	Excel includes features to make explicit queries to remote files. The attacker can select an option to get external data and set up a query to a remote text file. The text file should be written with tab spaces between words to specify different fields in the spreadsheet. By right-clicking on the cell and selecting Data Range Properties, the attacker can configure the query to update on open or even regularly (in the background).

Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.1: Dynamic content inclusion → D2-CAT2.1.1.3: Linked content
Target(s):	D3-CAT5: Information → D3-CAT5.1: Document D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Avoid the inclusion of dynamic content in the document to be signed

Name:	Substitution of Office document content by means of fields
Source:	Electronic Documents and Digital Signatures [131]
Description:	Several attacks can be performed using the field feature in some Office formats, like Word or Excel. Fields like TIME, USERNAME, etc. can make the visualization of a document content vary according to conditions controlled by the attacker. For instance, depending on the date when a document is opened or the user that opens the document, a piece of text can take one of several different possibilities. The content dependent on a field can be updated automatically in certain versions of Microsoft Word or explicitly via a macro.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.1: Dynamic content inclusion → D2-CAT2.1.1.1: Hidden code
Target(s):	D3-CAT5: Information → D3-CAT5.1: Document D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Avoid the inclusion of dynamic content in the document to be signed

Name:	Substitution of PDF content by means of javascript
Source:	Electronic Documents and Digital Signatures [131]
Description:	The attacker can use the form toolbar to create a form field, and then add Javascript code in its calculate field to change the value of the field according to the date.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.1: Dynamic content inclusion → D2-CAT2.1.1.2: Active code
Target(s):	D3-CAT5: Information → D3-CAT5.1: Document D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor

Countermeasures:	Avoid the inclusion of dynamic content in the document to be signed

Name:	Modification of HTML email content via Javascript
Source:	Electronic Documents and Digital Signatures [131]
Description:	An attack that modifies the content of an email formatted as HTML is performed by using the *document.write()* Javascript function and the current date.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.1: Dynamic content inclusion → D2-CAT2.1.1.2: Active code
Target(s):	D3-CAT5: Information → D3-CAT5.1: Document D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Avoid the inclusion of dynamic content in the document to be signed

Name:	Modification of HTML email content via embedded image
Source:	Electronic Documents and Digital Signatures [131]
Description:	The attacker embeds an image in a HTML formatted email and, in conjunction with Javascript, is able to modify the visualized content of the signed email.
Goal:	D1-CAT1: Deceive the signer to sign a document different than the intended one or under unintended conditions
Method:	D2-CAT2: Modification prior to signature computation → D2-CAT2.1: Document modification → D2-CAT2.1.1: Dynamic content inclusion → D2-CAT2.1.1.2: Active code
Target(s):	D3-CAT5: Information → D3-CAT5.1: Document D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Avoid the inclusion of dynamic content in the document to be signed

Name:	Signature creation data retrieval from low-security keys
Source:	Keyjacking: the surprising insecurity of client-side SSL [164]
Description:	Internet Explorer Web browser relies on Windows keystore and Cryptographic Service Provider (CSP) to store the private keys imported therein. Microsoft's CSP publishes a function called *CryptExportKey* which permits to directly obtain the private key from a keystore. A low-security key, which is the configuration by default, is a key imported in Internet Explorer which is not password-protected. Consequently, and based on previous facts, an attacker that gains access to the user's account or is able to execute malicious code with the user's privileges will the able to

	access the private key. The attacker could even export the private key for further usages.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.3: Unauthorized access to the SCDev → D2-CAT5.3.2: Authentication Bypass
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.4: SCDev
Countermeasures:	Use stronger configuration settings

Name:	Use of low-security keys
Source:	Keyjacking: the surprising insecurity of client-side SSL [164]
Description:	This attack is based on the same motivation as the attack *Signature creation data retrieval from low-security keys*. However, in this case the attacker does not retrieve the signature creation data but just performs as many signatures as desired without the user consent and knowledge. This attack is an alternative if the key was set to non-exportable.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.2: Authentication Bypass
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.4: SCDev
Countermeasures:	Use stronger configuration settings

Name:	Signature creation data retrieval from exportable medium-security keys
Source:	Keyjacking: the surprising insecurity of client-side SSL [164]
Description:	When a medium-security key is to be accessed (for signing or export), a warning is shown to the user, who must confirm the operation. This attack captures the warning event, hiding it to the user, during the key export operation (the key must be set as exportable). To achieve that, the attacker performs an API hijacking in which a function call made by the Internet Explorer process to the system Windows CryptoAPI is intercepted by a malicious DLL previously injected via a Windows Hook. Thereby, the attacker is able to hijack the call which displays the warning window, disabling it.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.3: Unauthorized access to the SCDev → D2-CAT5.3.2: Authentication Bypass
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.4: SCDev
Countermeasures:	-

Name:	Use of medium-security keys
Source:	Keyjacking: the surprising insecurity of client-side SSL [164]
Description:	This attack applies the same strategy as the attack*Signature creation data retrieval from exportable medium-security keys*. However, in this case the attacker does not retrieve the signature creation data but just performs as many signatures as desired without the user consent and knowledge. This attack is an alternative if the key was set to non-exportable.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.2: Authentication Bypass
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.4: SCDev
Countermeasures:	-

Name:	Signature creation data retrieval from high-security keys (in unrecommended configuration)
Source:	Keyjacking: the surprising insecurity of client-side SSL [164]
Description:	A high-security key requires the user to enter the associated password (SAD) each time the key is to be used or exported. However, if the user checked the box marked “Remember password”, the level of the key is downgraded to low-security, enabling the attacker to perform the same attack as in *Signature creation data retrieval from low-security keys*.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.3: Unauthorized access to the SCDev → D2-CAT5.3.2: Authentication Bypass
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.4: SCDev
Countermeasures:	Do not select “remember password” in the configuration settings

Name:	Use of high-security keys (in unrecommended configuration)
Source:	Keyjacking: the surprising insecurity of client-side SSL [164]
Description:	The attacker makes use of the same highly unrecommended configuration as in the attack *Signature creation data retrieval from high-security keys (in unrecommended configuration)*, being able to perform the same attack as in *Use of low-security keys*. This attack is an alternative if the key was set to non-exportable.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.2: Authentication Bypass
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.4: SCDev
Countermeasures:	Do not select “remember password” in the configuration settings

Name:	Signature creation data retrieval from exportable high-security keys
Source:	Keyjacking: the surprising insecurity of client-side SSL [164]
Description:	The attack is based on the same strategy as in *Signature creation data retrieval from exportable medium-security keys.* In this case, the attacker captures the invocation to the function that shows a window asking for a password each time the key is to be used. Once obtained the first time, the attacker is able retrieve the private key for further usages.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) $\rightarrow$ D2-CAT5.3: Unauthorized access to the SCDev $\rightarrow$ Compromise of the signer authentication data (SAD) $\rightarrow$ D2-CAT4.1.2: SAD interception $\rightarrow$ D2-CAT4.1.2.2: Interception in interprocess/entities communication
Target(s):	D3-CAT2: Software $\rightarrow$ D3-CAT2.1: Application $\rightarrow$ D3-CAT2.1.2: Related application $\rightarrow$ D3-CAT2.1.2.3: CSP
Countermeasures:	The authors indicate that there is no countermeasure for this security issue

Name:	Use of high-security keys
Source:	Keyjacking: the surprising insecurity of client-side SSL [164]
Description:	In case the key is set as non-exportable, the attacker can following the same actions as in *Signature creation data retrieval from exportable high-security keys* to compromise the access password.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function $\rightarrow$ D2-CAT4.1: Compromise of the signer authentication data (SAD) $\rightarrow$ D2-CAT4.1.2: SAD interception $\rightarrow$ D2-CAT4.1.2.2: Interception in interprocess/entities communication
Target(s):	D3-CAT2: Software $\rightarrow$ D3-CAT2.1: Application $\rightarrow$ D3-CAT2.1.2: Related application $\rightarrow$ D3-CAT2.1.2.3: CSP
Countermeasures:	The authors indicate that there is no countermeasure for this security issue

Name:	Use of high-security keys during the same session
Source:	Keyjacking: the surprising insecurity of client-side SSL [164]
Description:	This attack relies on a vulnerability by design in the CryptoAPI. In the context of Internet Explorer Web browser, once the CryptoAPI has authenticated a user when accessing a high-security key, subsequent accesses fail to request for the password. Using a malicious code that makes the same sequence of calls to the CryptoAPI as Internet Explorer, the attacker can perform as many signing operations as desired once the password has been provided by the user, and providing that the browser is not restarted.

Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.2: Authentication Bypass
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.4: SCDev
Countermeasures:	Close the Web browser once the desired operation is performed. Clear the SSL State in the Web browser configuration

Name:	Use of keys stored in cryptographic tokens
Source:	Keyjacking: the surprising insecurity of client-side SSL [164]
Description:	The attack applies the same strategy as in *Use of high-security keys* for keys stored in a particular external cryptographic token.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.1: Compromise of the signer authentication data (SAD) → D2-CAT4.1.2: SAD interception → D2-CAT4.1.2.2: Interception in interprocess/entities communication
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.3: CSP
Countermeasures:	-

Name:	Deception to use keys stored on cryptographic tokens
Source:	Keyjacking: the surprising insecurity of client-side SSL [164]
Description:	This attack makes use of social behavior to perform signatures on behalf of the user without his consent and knowledge. When a cryptographic token such as the Spanish electronic Identity Card (eDNI), Spyrus Rosetta USB and many others requests to user to insert the PIN or password in every access to the private key or protected areas of the internal file system, the user gets used to insert the credentials several times for a single operation (i.e. authenticate in a Web site, sign a document, etc.). The attacker will just request the user to enter the SAD in the middle of a normal operation or unexpectedly, and there will be a non-negligible probability for the user to do that.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.3: Unauthorized access to the SCDev → D2-CAT5.3.1: Compromise of the signer authentication data (SAD) → D2-CAT4.1.1: Social engineering
Target(s):	D3-CAT4: Human user → D3-CAT4.1: Signer
Countermeasures:	Apply a different design where the SAD is not required so many times. Caching the SAD during a single operation may lead to the attack *Using cached SAD to perform malicious signatures*

Name:	Using cached SAD to use keys stored on cryptographic tokens

Source:	Keyjacking: the surprising insecurity of client-side SSL [164]
Description:	In this attack, the attacker can perform as many signatures as desired if the CSP of a cryptographic token is configured to use the key for a specified time interval without asking for permission.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.2: Authentication Bypass
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.3: CSP
Countermeasures:	Applying a different design where the SAD is required for every single access to the key may lead to the attack *Deception to use keys stored on cryptographic tokens*

Name:	Signature creation data retrieval from password-protected files
Source:	BreakMS - Break Microsoft Private Key Encryption with a dictionary attack [88, 89]
Description:	This attack exploits several design and implementation vulnerabilities found in PKCS12 / PFX file format to perform a low-cost dictionary attack to discover the password used to protect the file and further retrieve the private key.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.3: Unauthorized access to the SCDev → D2-CAT5.3.1: Compromise of the signer authentication data (SAD) → D2-CAT4.1.3: Guessing
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.4: SCDev
Countermeasures:	Redesign and careful implementation of PKCS12 / PFX format

Name:	Unauthorized usage of platform resources by a malicious Applet in a Java-enabled card
Source:	Software attacks on smart cards [84]
Description:	If the Java Card where the Applet is loaded does not implement an access controller, then a Trojan horse embedded in the Applet can perform malicious operations. If there is no domain separation between simultaneous applets, the malicious one could extract sensitive information managed by another, like the PIN code, or modify critical data like the number of authentication attempts. These attacks could later derive in signature forgeries.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.1: Compromise of the signer authentication data (SAD) → D2-CAT4.1.2: SAD interception → D2-CAT4.1.2.2: Interception in interprocess/entities communication

Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.1: External application → D3-CAT2.1.1.2: User level application
Countermeasures:	Correct design and implementation of Java Card, specially Java Virtual Machine. Correct design and implementation of applets. Use of access controller. Use of shareable interfaces between applets (domain separation enforcement). More tips can be found in [84]

Name:	PIN phishing and Fraudulent signatures
Source:	Vulnerabilities of PKI based Smartcards [61]
Description:	The attacker reads the signer's authentication data (PIN of the smart card) entered by the user in the keyboard by means of a keylogger. Once the attacker has compromised the SAD, it is able to access the signing function of a smart card without the user knowing.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.1: Compromise of the signer authentication data (SAD) → D2-CAT4.1.2: SAD interception → D2-CAT4.1.2.2: Interception in interprocess/entities communication
Target(s):	D3-CAT2: Software → D3-CAT2.2: Driver → D3-CAT2.2.1: Keyboard driver
Countermeasures:	Use of secure I/O between the user and the Java Card: PIN entry from a cellular phone; separate hardware channel between the PKI card and a special I/O device that handles the user inputs; match-on-cards with own display

Name:	Remote control of PKI Card
Source:	Vulnerabilities of PKI based Smartcards [61]
Description:	An attacker is able to remotely request signing operations on the smart card once the user has unlocked it.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.2: Authentication Bypass
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.3: CSP
Countermeasures:	-

Name:	Timing Analysis attack in controlled environments
Source:	Timing attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems [138]
Description:	This was the first designed timing attack, which implementations were successful against Diffie-Hellman, RSA and DSS cryptosystems.

	These attacks were carried out in an isolated computing environment where the measured time could not be masked by delays provoked by processes running in the background.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.1: Timing Analysis
Target(s):	D3-CAT3: Hardware → D3-CAT3.1: SSCDev
Countermeasures:	Adapted blinding signatures can prevent attackers from knowing the input to the modular exponentiation function, with only low performance decrease [138]

Name:	Timing Analysis attack using the Chinese Remainder Theorem
Source:	A Timing Attack against RSA with the Chinese Remainder Theorem [208]
Description:	In this attack, a RSA-modulus is factorized providing that the exponentiation with the secret exponent uses the Chinese Remainder Theorem and Montgomery's algorithm.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.1: Timing Analysis
Target(s):	D3-CAT3: Hardware → D3-CAT3.1: SSCDev
Countermeasures:	Idem.

Name:	Remote Timing Analysis attack
Source:	Remote Timing Attacks are Practical [41]
Description:	Brumley and Boneh showed that remote attacks on real applications over a local network and running in general software systems are possible. In this case, they devised a timing attack against OpenSSL, guessing the private key used by the Web server for authenticating itself during the SSL handshake stage. This has been quite an important research since timing attacks are now possible although noisy intermediate elements such as network routers and background processes interact during the attack.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.1: Timing Analysis
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.4: SCDev
Countermeasures:	Enable the blinding feature of OpenSSL

Name:	Improved Remote Timing Analysis attack

Source:	Improving Brumley and Boneh Timing Attack on Unprotected SSL Implementations [4]
Description:	The authors improve the *Remote Timing Analysis attack* efficiency by a factor of more than ten. In particular, the attack exploit the timing behavior of Montgomery multiplications in the table initialization phase, which increases the number of multiplications that provide useful information to reveal one of the prime factors of RSA moduli.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.1: Timing Analysis
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.4: SCDev
Countermeasures:	Enable the blinding feature of OpenSSL

Name:	Simple Power Analysis attack (SPA)
Source:	Differential Power Analysis [139]
Description:	This type of power analysis attack is imperceptible to the user and can be successfully performed by using simple and cheap equipments. It only needs one or few measurements of power consumption signals to retrieve the private key stored in the cryptographic device.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.3: Power Analysis
Target(s):	D3-CAT3: Hardware → D3-CAT3.1: SSCDev
Countermeasures:	Make the power consumption of the cryptographic device independent of the signal values at the internal circuit nodes by either randomizing or flattening the power consumption. However, these techniques do not assure the device to be completely secure against these attacks, and instead they increase the required number of measurements [225]. If the attacker has access to the device for performing an enough number of operations, these countermeasures are useless.

Name:	Mono-bit Differential Power Analysis attack (DPA)
Source:	Differential Power Analysis [139] and An overview of side channel analysis attacks [147]
Description:	This type of power analysis attack is a statistical approach that examines a large number of power consumptions signals to retrieve secret keys. In particular, the mono-bit DPA analyzes the intermediate values of one bit.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)

Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.3: Power Analysis
Target(s):	D3-CAT3: Hardware → D3-CAT3.1: SSCDev
Countermeasures:	Idem.

Name:	Multi-bit Differential Power Analysis attack (DPA)
Source:	Ways to Enhance DPA [32] and An overview of side channel analysis attacks [147]
Description:	The difference between this attack and *Mono-bit Differential Power Analysis attack (DPA)* is that the former analyzes intermediate values of a set of several bits.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.3: Power Analysis
Target(s):	D3-CAT3: Hardware → D3-CAT3.1: SSCDev
Countermeasures:	Idem.

Name:	First-order Differential Power Analysis attack (DPA)
Source:	Differential Power Analysis [139]
Description:	In this case, the samples are observed at one instant of time.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.3: Power Analysis
Target(s):	D3-CAT3: Hardware → D3-CAT3.1: SSCDev
Countermeasures:	Idem.

Name:	High-order Differential Power Analysis attack (DPA)
Source:	On Second-Order Differential Power Analysis [127]
Description:	Contrary to *First-order Differential Power Analysis attack (DPA)*, this type of DPA attack analyzes the power consumption signals at some instants of time.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.3: Power Analysis
Target(s):	D3-CAT3: Hardware → D3-CAT3.1: SSCDev
Countermeasures:	Idem.

Name:	Correlation Power Analysis attack (CPA)
Source:	A proposition for Correlation Power Analysis enhancement [148] and An overview of side channel analysis attacks [147]

Description:	This type of attack consists of a technique based on the correlation between the real power consumption of the device and a certain power consumption model. DPA and CPA are based on power consumption models, so their efficiency completely depends on the chosen model. In case of wrongly modeling the power consumption, the key obtaining is impossible. Besides, these attacks need a large number of samples, and hence are not very practical.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.3: Power Analysis
Target(s):	D3-CAT3: Hardware → D3-CAT3.1: SSCDev
Countermeasures:	Idem.

Name:	Template Power Analysis attack
Source:	IPA: A New Class of Power Attacks [77], Template Attacks [48] and An overview of side channel analysis attacks [147]
Description:	This type of attack needs a reference device for executing a profiling stage. In this stage, a large number of signals are obtained from the reference device in order to learn how it works. During the second stage, the key extraction stage, the key is obtained by analyzing very few signals from the attacked device, improving the applicability of the attack respecting other types of power analysis attacks, like DPA or CPA. The reference device must be identical or very closed to the attacked device for the attack to work.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.3: Power Analysis
Target(s):	D3-CAT3: Hardware → D3-CAT3.1: SSCDev
Countermeasures:	Idem.

Name:	Stochastic Power Analysis attack
Source:	A Stochastic Model for Differential Side Channel Cryptanalysis [209] and An overview of side channel analysis attacks [147]
Description:	This attack needs a reference device like the *Template Power Analysis attack*. This attack uses a different strategy than the template-based attack. For instance, during the profiling stage, the power consumption is estimated by predefined functions, not from actual measured signals.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.3: Power Analysis

Target(s):	D3-CAT3: Hardware → D3-CAT3.1: SSCDev
Countermeasures:	Idem.

Name:	Electromagnetic Emanation attack on RSA
Source:	ElectroMagnetic Analysis (EMA): Measures and Countermeasures for Smart Cards [195] and Electromagnetic Analysis: Concrete Results [80]
Description:	An attack to an RSA implementation was successfully carried out, focusing on the RSA modular exponentiation performed in a decapsulated smart card.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.2: Electromagnetic Analysis
Target(s):	D3-CAT3: Hardware → D3-CAT3.1: SSCDev
Countermeasures:	Hardware countermeasures: metal layer addition to the chip; active grid placement on top of the chip, in order to introduce more noise into the EM field, blurring the emanations [162]

Name:	Electromagnetic Emanation attack by using the channel capacity information
Source:	Evaluation of Information Leakage via Electromagnetic Emanation and Effectiveness of Tempest [223]
Description:	In this study, it is shown how to estimate the amount of information leakage by using the value of channel capacity, that it, the communication channel between the measured IT device and the receiver. This IT device can be both a personal computer or a smart card.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.2: Electromagnetic Analysis
Target(s):	D3-CAT3: Hardware → D3-CAT3.1: SSCDev D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.4: SCDev
Countermeasures:	Idem.

Name:	A low cost Electronic Emanation attack on a smart card
Source:	Low cost attacks on smart cards: The electromagnetic side-channel[162]
Description:	With this attack, it is demonstrated that performing EMA attacks using limited technical knowledge as well as cheap resources is possible. EM traces are successfully acquired from the sample card, and an analysis software correctly identifies the key.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)

Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.2: Electromagnetic Analysis
Target(s):	D3-CAT3: Hardware → D3-CAT3.1: SSCDev
Countermeasures:	Idem.

Name:	Fault-based attack on RSA
Source:	Fault-Based Attack of RSA Authentication [183]
Description:	In this paper, a theoretical systematic fault-based attack on the modular exponentiation algorithm for RSA is developed. Later on, the authors carry out a practical and complete end-to-end fault-attack on a microprocessor system, exploiting the vulnerabilities of an FPGA implementation of the system under attack and which runs a flawed OpenSSL software implementation. The authors inject transient faults in the target machine by regulating the voltage supply of the system, not requiring access to the system's internal components but just proximity to it. The authors are able to extract the 1024-bit RSA private key in approximately 100 hours.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.4: Microarchitectural Analysis
Target(s):	D3-CAT3: Hardware → D3-CAT3.1: SSCDev D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.4: SCDev
Countermeasures:	

Name:	A Branch Prediction Analysis attack on RSA: Exploiting the Predictor directly (Direct Timing Attack)
Source:	Predicting Secret Keys via Branch Prediction [3]
Description:	This is a type of microarchitectural side-channel attack called branch prediction analysis (BPA) attack, by which the branch prediction capability, common to all modern high-performance CPUs, is exploited to know the private key used in a software cryptographic algorithm. In particular, the penalty payed (extra clock cycles) for a mispredicted branch can be used for cryptanalysis of cryptographic primitives that employ a data-dependent program flow. This attack relies on the fact that the prediction algorithms are deterministic, and assume that the RSA implementation employs Square-and-Multiply exponentiation and Montgomery Multiplication. Though this attack is experimentally carried out on a simple RSA implementation, the underlying ideas can be used to develop similar attacks on different implementations of RSA and/or on other ciphers based upon ECC.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)

Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.4: Microarchitectural Analysis
Target(s):	D3-CAT2: Software → D3-CAT2.3: Operating system
Countermeasures:	-

Name:	A Branch Prediction Analysis attack on RSA: Forcing the BPU to the Same Prediction (Asynchronous Attack)
Source:	Predicting Secret Keys via Branch Prediction [3]
Description:	In this attack it is assumed that the cipher runs on a simultaneous multi-threading computer. The attacker can run a dummy process simultaneously with the cipher process, but the two parallel threads are isolated and share only the common Branch Prediction Unit (BPU) resource. Also, the attacker does not need to know any detail of the prediction algorithm., like in the previous attack.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.4: Microarchitectural Analysis
Target(s):	D3-CAT2: Software → D3-CAT2.3: Operating system
Countermeasures:	-

Name:	A Branch Prediction Analysis attack on RSA: Forcing the BPU to the Same Prediction (Synchronous Attack)
Source:	Predicting Secret Keys via Branch Prediction [3]
Description:	In this attack, the malicious process needs some sort of synchronization with the simultaneous crypto-process. It is also assumed that the RSA implementation employs Square-and-Multiply exponentiation. Any implementation of a cryptosystem is vulnerable to this kind of attack if the execution flow is key-dependent, including several implementations that had been considered to be immune to certain types of of side-channel attacks.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.4: Microarchitectural Analysis
Target(s):	D3-CAT2: Software → D3-CAT2.3: Operating system
Countermeasures:	-

Name:	A Branch Prediction Analysis attack on RSA: Trace-driven Attack against the BTB (Asynchronous Attack)
Source:	Predicting Secret Keys via Branch Prediction [3]

Description:	In this attack, it is assumed that the attacker can run a spy process simultaneously with the cipher, but it does not need to be synchronized with it. The same cryptographic implementations vulnerable to the previous attack are vulnerable to this one. Furthermore, this attack is much easier to be put in practice, and, in the authors' opinion, this attack puts many of the current public-key implementations in danger.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.4: Microarchitectural Analysis
Target(s):	D3-CAT2: Software → D3-CAT2.3: Operating system
Countermeasures:	-

Name:	A Simple Branch Prediction Analysis attack on RSA
Source:	On the Power of Simple Branch Prediction Analysis [2]
Description:	This is a a BPA variation by which almost all of the RSA key bits can be extracted during a single RSA operation.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.4: Microarchitectural Analysis
Target(s):	D3-CAT2: Software → D3-CAT2.3: Operating system
Countermeasures:	-

Name:	An Instruction Cache Analysis attack on the RSA implementation of OpenSSL
Source:	Yet another MicroArchitectural Attack: Exploiting I-cache [1]
Description:	This attack exploits the behavior of the Instruction Cache - which is used to reduce the average time to read instruction codes from main memory - to extract sensitive information regarding the execution of a cryptosystem. More specifically, this attack targets the OpenSSL sliding Window exponentiation of its RSA implementation.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.2: Eavesdropping (side-channel) → D2-CAT5.2.4: Microarchitectural Analysis
Target(s):	D3-CAT2: Software → D3-CAT2.3: Operating system
Countermeasures:	-

Name:	PIN/Password recovering from keyboard acoustic emanations
Source:	Keyboard Acoustic Emanations Revisited [239]

Description:	The authors built a prototype that can bootstrap a keyboard acoustic recognizer from about 10 minutes of English text typing, using about 30 minutes of computation on an average desktop computer. After that, the prototype can recognize keystrokes in real time, including random ones such as passwords, with an accuracy rate of about 90%. The keystrokes must be typed by the same person, with the same keyboard, under the same recording conditions. These conditions can easily be satisfied by, for example, placing a wireless microphone in the user's work area or by using parabolic microphones. This attack could be mounted to compromise the signer's authentication data. As a result, it would be the earliest stage before accessing the signature creation data or the signing function. As such, this partial attack can be classified under the two methods indicated below.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.1: Compromise of the signer authentication data (SAD) → D2-CAT4.1.3: Guessing D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.3: Unauthorized access to the SCDev → D2-CAT5.3.1: Compromise of the signer authentication data (SAD) → D2-CAT4.1.3: Guessing
Target(s):	D3-CAT3: Hardware → D3-CAT3.2: Computer → D3-CAT3.2.4: Peripheral devices → D3-CAT3.2.4.2: Keyboard
Countermeasures:	Ensure the physical security of the machine and the room. Use of two-factor authentication (e.g. password and biometrics) to access the signature creation data.

Name:	Finding collisions in several MD3,MD5, HAVAL, RIPEMD and SHA-0
Source:	How to Break MD5 and Other Hash Functions [231]
Description:	A new differential attack on several hash functions is described. The attack, called modular differential, unlike most differential attacks, uses modular integer subtraction as the measure instead of the exclusive-or. In the case of MD3, the attack can find a collision within less than a second, and can also find second preimages for many messages. For MD5, it finds collisions in about 15 minutes up to an hour computation time. As the attack can be carried out following two different methods (collision or second pre-image), the method of the attack could be classified attending to both approaches.
Goal:	D1-CAT3: Replace signed information
Method:	D2-CAT3: Modification post signature computation → D2-CAT3.2: Cryptanalysis → D2-CAT3.2.1: Hash function → D2-CAT3.2.1.1: Collision attack
Target(s):	D3-CAT1: Cryptography
Countermeasures:	Use stronger hash functions.

Name:	Finding MD5 collisions using tunnels
Source:	Tunnels in hash functions: MD5 collisions within a minute [137]
Description:	The author proposes a new strategy to find collisions in hash functions named tunneling. Tunnels replace multi-message modification methods and exponentially accelerate collision search. In particular, the author describe several tunnels in hash function MD5. By using them, a MD5 collision is found in approximately one minute on a standard notebook PC (Intel Pentium, 1.6 GHz). This attack is a collision attack, since it finds two messages which hash coincides. The method works for any initializing value. For this attack to succeed, the attacker must trick the user to sign one of the messages (possibly the message is aligned with the user's interests), and afterwards replace it by the fraudulent one (see birthday attack [58])
Goal:	D1-CAT3: Replace signed information
Method:	D2-CAT3: Modification post signature computation → D2-CAT3.2: Cryptanalysis → D2-CAT3.2.1: Hash function → D2-CAT3.2.1.1: Collision attack
Target(s):	D3-CAT1: Cryptography
Countermeasures:	Use stronger hash functions.

Name:	Using Expandable Messages to Find Second Preimages
Source:	Second preimages on n-bit hash functions for much less than 2^n work [134]
Description:	The authors describe a generic way to carry out long-message second preimage attacks, despite the Damgard-Merkle strengthening done on all modern hash functions (including SHA-1). The work required to achieve the attack is substantially lower than the reference one (2^n). For instance, using SHA-1 as an example, the attack can find a second preimage for a 2^{60} byte message in 2^{106} work, rather than the previously expected 2^{160} work. Though the attack is theoretical (e.g. the messages for which second preimages may be found are generally impractically long), the authors showed that an n-bit iterated hash function cannot provide the expected second-preimage resistance for long messages. As a second preimage attack, the attacker would be able to compose a malicious document which hash value matched the one of the signed document.
Goal:	D1-CAT3: Replace signed information
Method:	D2-CAT3: Modification post signature computation → D2-CAT3.2: Cryptanalysis → D2-CAT3.2.1: Hash function → D2-CAT3.2.1.3: Second preimage attack
Target(s):	D3-CAT1: Cryptography
Countermeasures:	Use stronger hash functions.

Name:	Herding attack on hash functions
Source:	Herding Hash Functions and the Nostradamus Attack [133]

Description:	The authors define a property of a hash function, Chosen Target Forced Prefix (CTFP) preimage resistance, which is both important for real-world applications of hash functions, and dependent on collision resistance of the hash function. More specifically, the described attack, called the herding attack, affects Damgard-Merkle hash functions in a way that the attacker who can find many collisions on the hash function by brute force can first provide the hash of a message, and later "herd" any given starting part of a message (P) to that hash value by the choice of an appropriate suffix (S). This attack can be considered a practical improvement of *Using Expandable Messages to Find Second Preimages* where the resulting message can be of a reasonable size. The authors provide concrete examples of carrying out the attack. One of them, named *Tweaking a Signed Document*, considers the case where a signer can later produce a modified message while still resulting in the same hash. As stated by the authors, many applications of hashing for signatures which are not vulnerable to attack by straightforward collision-finding techniques are broken by an attacker who can violate CTFP preimage resistance. When the CTFP definition is relaxed somewhat the attacks become still cheaper and more practical. For instance, if the attacker has control over the format of P - easy if the attacker intercepts the document to be signed, giving him prior knowledge of the full (large) set of possible P strings that might be presented (this is possible in certain transactions where the skeleton of the DTBS is fixed and just few parts of the document can vary). This is a preimage attack since the attacker manipulates part of the data entered in the hash function in order to obtain the desired hash value.
Goal:	D1-CAT3: Replace signed information
Method:	D2-CAT3: Modification post signature computation $\rightarrow$ D2-CAT3.2: Cryptanalysis $\rightarrow$ D2-CAT3.2.1: Hash function $\rightarrow$ D2-CAT3.2.1.2: Preimage attack
Target(s):	D3-CAT1: Cryptography
Countermeasures:	Use stronger hash functions.

Name:	Preimage attack on RIPEMD
Source:	Preimage Attack on Hash Function RIPEMD [230]
Description:	The first preimage attack on the RIPEMD hash function is described. Three variants are shown: an attack on the compression function of the 26-step reduced RIPEMD, with complexity 2^{110} compression function computations; an attack on the 26-step reduced RIPEMD with complexity $2^{115.2}$ instead of 2^{128}; and an attack on 29 steps with the same complexity. Furthermore, the complexity of the preimage attack on the full RIPEMD without the padding rule is reduced to 2^{127}, which optimizes the complexity order to brute-force attack.
Goal:	D1-CAT3: Replace signed information

Method:	D2-CAT3: Modification post signature computation → D2-CAT3.2: Cryptanalysis → D2-CAT3.2.1: Hash function → D2-CAT3.2.1.2: Preimage attack
Target(s):	D3-CAT1: Cryptography
Countermeasures:	Use stronger hash functions.

Name:	Parallel RSA factorization using the Multiple Polynomial Quadratic Sieve (MPQS)
Source:	A Study on Parallel RSA Factorization [235]
Description:	In this paper, a factorization of a 100-digit RSA modulus into the former primer numbers is presented. The experimental result shows that it takes 6.6 days for factoring the 100-digit number using the enhanced MPQS by 32 workstations.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.4: Cryptanalysis → D2-CAT5.4.1: Asymmetric algorithm
Target(s):	D3-CAT1: Cryptography
Countermeasures:	Use large RSA key lengths (currently recommended 1024 bits and above).

Name:	Integer factorization with TWINKLE
Source:	Analysis and optimization of the TWINKLE factoring Device [153]
Description:	TWINKLE (The Weizmann Institute Key Locating Engine) is an optoelectronic device designed to be capable of factoring large integers by speeding up the sieving step of the Quadratic Sieve and Number Field Sieve factoring algorithms. The authors consider that a TWINKLE-assisted factorization of a 768-bit number is feasible in about 9 months using a set of 80.000 standard Pentium II PC's and 5.000 TWINKLE devices. The advances in computers since 2000 let us foresee that the time needed to factoring large numbers would imply a bound lower than 9 months.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.4: Cryptanalysis → D2-CAT5.4.1: Asymmetric algorithm
Target(s):	D3-CAT1: Cryptography
Countermeasures:	Use large RSA key lengths (currently recommended 1024 bits and above).

Name:	Integer factorization with TWIRL
Source:	Special-Purpose Hardware for Factoring: the NFS Sieving Step [212]

Description:	As the authors comment, it is commonly claimed that 1024-bit RSA keys are safe in a medium term (15 years, maybe more), since when applying the Number Field Sieve (NFS) to such composites both the sieving step and the linear algebra step would be unfeasible. However, the introduction of special-purpose hardware architectures for NFS, like TWINKLE or TWIRL, has reduced the predicted cost of factoring 1024-bit numbers by several orders of magnitude. The authors estimate that factoring a 1024-bit integer using TWIRL - the evolution of TWINKLE (see *Integer factorization with TWINKLE*) - would be possible in one year at the cost of a few dozen million US dollars.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.4: Cryptanalysis → D2-CAT5.4.1: Asymmetric algorithm
Target(s):	D3-CAT1: Cryptography
Countermeasures:	Use even larger RSA key lengths (2048 or 4096 bits).

Name:	Signature application substitution
Source:	This thesis.
Description:	This kind of attack tries to compromise sensitive data by replacing the SCA by a fake one. If the user does not notice the difference, he will have completely felt into the hands of the attacker. Depending on the purpose of the attack and the nature of the SCD, the attacker would be able to compromise either the SAD or the SCD itself. As such, two methods of attacks are applied.
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.3: Unauthorized access to the SCDev → D2-CAT5.3.1: Compromise of the signer authentication data (SAD) → D2-CAT4.1.2: SAD interception → D2-CAT4.1.2.3: Endpoint compromise D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.1: SCD interception → D2-CAT5.1.2: Endpoint compromise
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.2: SCA
Countermeasures:	Verify the integrity of the software before installing it. Implement integrity verification routines (e.g. TPM) for critical software during start-up

Name:	SCD compromise during issuance
Source:	This thesis.
Description:	The SCD is exposed and can be intercepted by an attacker if the Certification Authority sends the SCD through an unprotected channel to the entity in charge of writing the SCD in the SSCDev.

Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT5: Compromise of the signature creation data (SCD) → D2-CAT5.1: SCD interception → D2-CAT5.1.1: Interception in interprocess/entities communication
Target(s):	D3-CAT2: Software → D3-CAT2.4: Network → D3-CAT2.4.1: Protocols
Countermeasures:	Use of protected channels

Name:	SAD compromise by shoulder surfing
Source:	Information Systems Security: A Practitioner's Reference [78].
Description:	The attacker observes the signer introducing the SAD in the Platform of the SCS (e.g. before generating a signature).
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.1: Compromise of the signer authentication data (SAD) → D2-CAT4.1.2: SAD interception → D2-CAT4.1.2.1: Observation
Target(s):	D3-CAT4: Human user → D3-CAT4.1: Signer
Countermeasures:	-

Name:	SAD compromise by optical emanation
Source:	Information Leakage from Optical Emanations [159].
Description:	The authors describe two implementations of a Trojan horse that manipulates the LEDs on a standard keyboard to implement a high-bandwidth covert channel. The attack can be mounted to obtain the information stored in the computer or typed by the user (e.g. the SAD).
Goal:	D1-CAT2: Unauthorized use of the signature creation data (SCD)
Method:	D2-CAT4: Unauthorized invocation of the signing function → D2-CAT4.1: Compromise of the signer authentication data (SAD) → D2-CAT4.1.2: SAD interception → D2-CAT4.1.2.1: Observation
Target(s):	D3-CAT3: Hardware → D3-CAT3.2: Computer → D3-CAT3.2.4: Peripheral devices → D3-CAT3.2.4.2: Keyboard
Countermeasures:	-

Name:	Font type manipulation - Fonts name change
Source:	What You See is Not Always What You Sign [130]
Description:	This attack improves *Font type manipulation - Fonts substitution* attack by using a customized font type renamed to the expected one. As a result, the verifier is not able to distinguish whether the computer where the signature was computed had a different font type installed.
Goal:	D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content

Method:	D2-CAT1: Environment manipulation
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Use of formats (e.g. PDF) that include the fonts definitions inside the content of the document

Name:	False positives in ASN.1
Source:	What You See is Not Always What You Sign [130]
Description:	If the verifier uses an ASN.1 encoding rules different than the certificate issuer, it permits an attacker to generate a signature with a revoked certificate without being detected in the CRLs.
Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result
Method:	D2-CAT6: Influence on certificate verification result → D2-CAT6.2: Alteration of certificate status verification → D2-CAT6.2.6: Alteration of certificate status verification result
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.5: SVA
Countermeasures:	Correct application of encoding rules

Name:	Secure viewer compromise for fraudulent signature verification (1)
Source:	Malware Attacks on Electronic Signatures Revisited [145]
Description:	The attack is carried out on Deutsche Telekom T-Telesec Signet 1.6.0.4 product. The attack does not need administrator privileges and relies on design flaws, not implementation ones. In particular, the attack modifies the viewer's presentation surface without detection to deceive the user respecting the result of the signature verification.
Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.1: Presentation manipulation → D2-CAT7.1.3: Verification result masquerading
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.5: SVA
Countermeasures:	-

Name:	Secure viewer compromise for fraudulent signature verification (2)
Source:	Malware Attacks on Electronic Signatures Revisited [145]
Description:	The attack is carried out on IT Solution trustDesk standard 1.2.0 product. The attack does not need administrator privileges and relies on design flaws, not implementation ones. In particular, the attack modifies the viewer's presentation surface without detection to deceive the user respecting the result of the signature verification.

Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.1: Presentation manipulation → D2-CAT7.1.3: Verification result masquerading
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.5: SVA
Countermeasures:	-

Name:	Secure viewer compromise for fraudulent signature verification (3)
Source:	Malware Attacks on Electronic Signatures Revisited [145]
Description:	The attack is carried out on D-Sign matrix/digiSeal 3.0.1 product. The attack does not need administrator privileges and relies on design flaws, not implementation ones. In particular, the attack modifies the viewer's presentation surface without detection to deceive the user respecting the result of the signature verification.
Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.1: Presentation manipulation → D2-CAT7.1.3: Verification result masquerading
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.5: SVA
Countermeasures:	-

Name:	Manipulated presentation of signed data for fraudulent verification
Source:	Malware Attacks on Electronic Signatures Revisited [145]
Description:	This attack violates the What-Is-Presented-Is-What-Is-Signed (WIPIWIS) principle. The attack is carried out on Ventasoft venta-sign 2.0.0.968 product. The attack does not need administrator privileges and relies on design flaws, not implementation ones. In particular, the attack modifies the application's presentation surface without detection to deceive the user respecting the signature verification and integrity checker software results.
Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.1: Presentation manipulation → D2-CAT7.1.1: DTBV masquerading → D2-CAT7.1.1.1: Document masquerading
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.5: SVA
Countermeasures:	-

Name:	Secure viewer compromise for fraudulent signature verification (4)

Source:	Malware Attacks on Electronic Signatures Revisited [145]
Description:	The attack is carried out on 2B Secure FILE 1.0 product. The attack does not need administrator privileges and relies on design flaws, not implementation ones. In particular, the attack modifies the viewer's presentation surface without detection to deceive the user respecting the result of the signature verification.
Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result
Method:	D2-CAT7: Influence on signature verification result $\rightarrow$ D2-CAT7.1: Presentation manipulation $\rightarrow$ D2-CAT7.1.3: Verification result masquerading
Target(s):	D3-CAT2: Software $\rightarrow$ D3-CAT2.1: Application $\rightarrow$ D3-CAT2.1.2: Related application $\rightarrow$ D3-CAT2.1.2.5: SVA
Countermeasures:	-

Name:	Secure viewer compromise for fraudulent signature verification (5)
Source:	Malware Attacks on Electronic Signatures Revisited [145]
Description:	The attack is carried out on Ultimaco SafeGuard Sign & Crypt for Office 3.4.1 product. The attack does not need administrator privileges and relies on design flaws, not implementation ones. In particular, the attack modifies the viewer's presentation surface without detection to deceive the user respecting the result of the signature verification.
Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result
Method:	D2-CAT7: Influence on signature verification result $\rightarrow$ D2-CAT7.1: Presentation manipulation $\rightarrow$ D2-CAT7.1.3: Verification result masquerading
Target(s):	D3-CAT2: Software $\rightarrow$ D3-CAT2.1: Application $\rightarrow$ D3-CAT2.1.2: Related application $\rightarrow$ D3-CAT2.1.2.5: SVA
Countermeasures:	-

Name:	Collisions in PDF Signatures
Source:	Collisions in PDF Signatures [240]
Description:	This attack violates the What-Is-Presented-Is-What-Is-Signed (WIPIWIS) principle. The author describes a vulnerability in the PDF standard. Using this vulnerability, an attacker is capable of producing a PDF document which is shown differently when opened, and due to the way the signature blob had been injected by the attacker. Therefore, two different (as shown) documents produce the same signature.
Goal:	D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content
Method:	D2-CAT7: Influence on signature verification result $\rightarrow$ D2-CAT7.1: Presentation manipulation $\rightarrow$ D2-CAT7.1.1: DTBV masquerading $\rightarrow$ D2-CAT7.1.1.1: Document masquerading

Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	-

Name:	Dali attack (verification)
Source:	The Dali Attack on Digital Signature [43]
Description:	This attack violates the What-Is-Presented-Is-What-Is-Signed (WIPIWIS) principle. Attack based on the capability of a file of having a static polymorphic behavior. The attacker prepares the signed document to include a secondary content. Thanks to certain formats tagging, the content shown to the verifier varies depending on the file extension, and thus the application chosen to open the file. The attack is limited to the inclusion of HTML as the malicious secondary content.
Goal:	D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.1: Presentation manipulation → D2-CAT7.1.2: Viewer manipulation → D2-CAT7.1.2.1: Viewer substitution
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Inclusion of the signed attribute *content-type* in the electronic signature format (e.g. CAdES, XAdES)

Name:	Enhanced Dali attack (verification)
Source:	Fortifying the Dali Attack on Digital Signature [44]
Description:	Attack that enhances the Dali Attack to permit the usage of tiff and PDF formats for the contents inserted in the signed document.
Goal:	D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.1: Presentation manipulation → D2-CAT7.1.2: Viewer manipulation → D2-CAT7.1.2.1: Viewer substitution
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Use of PDF/A formats. Use of PDF Advanced Electronic Signature (PAdES) formats. Inclusion of the signed attribute *content-type* in the electronic signature format (e.g. CAdES, XAdES)

Name:	Inconsistent handling of HTML table tags (verification)
Source:	What You See is Not Always What You Sign [130]

Description:	This attack violates the What-Is-Presented-Is-What-Is-Signed (WIPIWIS) principle. Web browsers interpret HTML and Javascript code in a different manner. Consequently, the same HTML code can be shown in different ways depending on the web browser used.
Goal:	D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.1: Presentation manipulation → D2-CAT7.1.1: DTBV masquerading → D2-CAT7.1.1.1: Document masquerading
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.1: External application → D3-CAT2.1.1.2: User level application
Countermeasures:	Detect the existence of dynamic content in the signed document

Name:	Substitution of Office document by external content using macros (verification)
Source:	Electronic Documents and Digital Signatures [131]
Description:	This attack violates the What-Is-Presented-Is-What-Is-Signed (WIPIWIS) principle. When opening the signed document, some active code (e.g. a macro programmed in Visual Basic for Applications for a Word document or an Excel spreadsheet) included in it substitutes the content of the document by an external content controlled by the attacker. This attack is feasible on Microsoft Office formats. As the signature is verified against the initial object, the signature integrity is not corrupted.
Goal:	D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.1: Presentation manipulation → D2-CAT7.1.1: DTBV masquerading → D2-CAT7.1.1.1: Document masquerading
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Detect the existence of dynamic content in the signed document

Name:	External queries in Excel (verification)
Source:	Electronic Documents and Digital Signatures [131]
Description:	This attack violates the What-Is-Presented-Is-What-Is-Signed (WIPIWIS) principle. Excel includes features to make explicit queries to remote files. The attacker can select an option to get external data and set up a query to a remote text file. The text file should be written with tab spaces between words to specify different fields in the spreadsheet. By right-clicking on the cell and selecting Data Range Properties, the attacker can configure the query to update on open or even regularly (in the background).
Goal:	D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content

Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.1: Presentation manipulation → D2-CAT7.1.1: DTBV masquerading → D2-CAT7.1.1.1: Document masquerading
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Detect the existence of dynamic content in the signed document

Name:	Substitution of Office document content by means of fields (verification)
Source:	Electronic Documents and Digital Signatures [131]
Description:	This attack violates the What-Is-Presented-Is-What-Is-Signed (WIPIWIS) principle. Several attacks can be performed using the field feature in some Office formats, like Word or Excel. Fields like TIME, USERNAME, etc. can make the visualization of a document content vary according to conditions controlled by the attacker. For instance, depending on the date when a document is opened or the user that opens the document, a piece of text can take one of several different possibilities. The content dependent on a field can be updated automatically in certain versions of Microsoft Word or explicitly via a macro.
Goal:	D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.1: Presentation manipulation → D2-CAT7.1.1: DTBV masquerading → D2-CAT7.1.1.1: Document masquerading
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Detect the existence of dynamic content in the signed document

Name:	Substitution of PDF content by means of javascript (verification)
Source:	Electronic Documents and Digital Signatures [131]
Description:	This attack violates the What-Is-Presented-Is-What-Is-Signed (WIPIWIS) principle. The attacker can use the form toolbar to create a form field, and then add Javascript code in its calculate field to change the value of the field according to the date.
Goal:	D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.1: Presentation manipulation → D2-CAT7.1.1: DTBV masquerading → D2-CAT7.1.1.1: Document masquerading
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Detect the existence of dynamic content in the signed document

Name:	Modification of HTML email content via Javascript (verification)

Source:	Electronic Documents and Digital Signatures [131]
Description:	This attack violates the What-Is-Presented-Is-What-Is-Signed (WIPIWIS) principle. An attack that modifies the content of an email formatted as HTML is performed by using the *document.write()* Javascript function and the current date.
Goal:	D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.1: Presentation manipulation → D2-CAT7.1.1: DTBV masquerading → D2-CAT7.1.1.1: Document masquerading
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Detect the existence of dynamic content in the signed document

Name:	Modification of HTML email content via embedded image (verification)
Source:	Electronic Documents and Digital Signatures [131]
Description:	This attack violates the What-Is-Presented-Is-What-Is-Signed (WIPIWIS) principle. The attacker embeds an image in a HTML formatted email and, in conjunction with Javascript, is able to modify the visualized content of the signed email.
Goal:	D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.1: Presentation manipulation → D2-CAT7.1.1: DTBV masquerading → D2-CAT7.1.1.1: Document masquerading
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.1: Document processor
Countermeasures:	Detect the existence of dynamic content in the signed document

Name:	Modification of the request of revocation of a compromised certificate to achieve successful fraudulent signature verification
Source:	This thesis.
Description:	The premise of this attack is that the attacker has compromised a private key with which he wants to sign a document on behalf of the legitimate owner. It is also assumed that the owner of the key has detected such compromise, and thus proceeds to revoke the corresponding certificate. In this potential attack, the revocation request is modified by the attacker before it is authenticated by the owner of the certificate. For the attack to be effective, the attacker must change the information of the request that identifies the certificate which revocation is being requested. As a result, the revocation will not become effective, and the verifier will conclude that the signature is valid.
Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result

Method:	D2-CAT6: Influence on certificate verification result → D2-CAT6.1: Alteration of subscriber's revocation request → D2-CAT6.1.2: Modification of revocation request
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.1: External application → D3-CAT2.1.1.2: User level application
Countermeasures:	-

Name:	Deny the revocation of a compromised certificate to achieve successful fraudulent signature verification
Source:	This thesis.
Description:	The premise of this attack is that the attacker has compromised a private key with which he wants to sign a document on behalf of the legitimate owner. It is also assumed that the owner of the key has detected such compromise, and thus proceeds to revoke the corresponding certificate. In this potential attack, the revocation request is intercepted by the attacker. If the revocation protocol does not incorporate a revocation response (e.g. as permitted by IETF CMP [6], the owner of the certificate will not notice whether the revocation reached the certification authority or not. As a result, the revocation will not become effective, and the verifier will conclude that the signature is valid.
Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result
Method:	D2-CAT6: Influence on certificate verification result → D2-CAT6.1: Alteration of subscriber's revocation request → D2-CAT6.1.1: DoS of revocation request
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.1: External application → D3-CAT2.1.1.2: User level application
Countermeasures:	-

Name:	Identity theft by untrusted trust anchor addition
Source:	This thesis.
Description:	In this potential attack, the attacker produces either a self-signed certificate or a certificate issued by a faked certification authority. This certificate contains the identity of the victim. Afterwards, the attacker compromises the trusted store of the verifier to inject the trust anchor that will allow a successful certification chain validation. Thereby, the attacker is able to sign documents masquerading as another entity (the victim), and the verifier will trust the fake certificate.
Goal:	D1-CAT4: Make the signed document be attributed to a user different than the actual signer
Method:	D2-CAT6: Influence on certificate verification result → D2-CAT6.3: Untrusted trust anchor/trust point addition
Target(s):	D3-CAT5: Information → D3-CAT5.3: Cryptographic material → D3-CAT5.3.1: Trust store
Countermeasures:	

Name:	Successful fraudulent signature verification by delaying the time-stamped signature sending
Source:	This thesis.
Description:	CEN CWA 14171 [47] establishes that the verifier, before assessing the validity of the certificate associated to the signature, should ascertain that at least the grace period has elapsed since a signature relevant time. The grace period is defined as the *time period which permits the certificate revocation information to propagate through the revocation process to relying parties; it is the minimum time period an initial verifier has to wait to allow any authorized entity to request a certificate revocation and the relevant revocation status provider to publish revocation status*. CEN CWA 14171 also indicates that the signature relevant time should be the time indicated in an associated TST or in an associated time mark. On the other hand, the cautionary period is defined at Certification Practices Statement level [50], which allows the legitimate owner of a digital certificate to withdraw the validity of a recently generated signature by revoking the corresponding certificate a posteriori, that is, once the signature has been computed. Assuming a delay between the time when a key is compromised and the time when the user notices it and requests the revocation of the corresponding certificate(s), the cautionary period offers the users a mechanism for preventing the attackers to benefit from the signatures performed during this time frame. The verifier should wait a period (the cautionary period) after receiving a signature to allow certificate revocation requests to be processed by the CA, even when these requests were made after the signature computation. In this situation, grace and cautionary periods mean the same concept. In this potential attack, it is being assumed that the legitimate owner of the certificate (user) cannot detect the private key compromise before the attacker makes use of the signed document and the corresponding signature. On the other hand, it is also assumed that the attacker cannot benefit from the signed document before the cautionary period expires, diminishing the attacker's chances. Section 5.2 of CEN CWA 14171 permits that a signer acts as an initial verifier as well, being capable of adding a trusted time-stamp or time-mark to the signature. Suppose that an attacker compromises a user's private key, signs a desired document with it and time stamps the generated signature. Let's consider that the user detects the key compromise once another entity, like the verifier, receives the signature. If an entity different to the attacker knows the existence of the signature, it is possible that the user is somehow notified about that (possibly during the grace period) and then he could proceed to request the certificate(s) revocation, preventing the attacker to benefit from the forged signature.

	However, if the attacker delays the signature sending until the CRL is updated, then the verifier will possess a CRL issued after the signing time (specified by the time-stamp), and will not wait for any further update. The CRL next update value can be easily guessed by the attacker just by taking a look at the 'nextUpdate' field of the CRL data structure [57]. As a result, and though made, the revocation request will have no effect. The signature will be considered valid and the attacker will be able to benefit from it although the certificate revocation is afterwards published. This attack could also be performed using time-marks.
Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result
Method:	D2-CAT6: Influence on certificate verification result → D2-CAT6.2: Alteration of certificate status verification → D2-CAT6.2.1: Grace or cautionary period bypassing → D2-CAT6.2.1.1: Delay in time-stamped signature sending
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.5: SVA
Countermeasures:	If the verifier receives a signature a long time after the time indicated in the time-stamp included in the signature by the signer, then the attack described herein could have been applied. A security policy should indicate whether the signature should be considered as invalid or not, depending on such elapsed time.

Name:	Successful fraudulent signature verification by exploiting the delay in CA's revocation request processing
Source:	This thesis.
Description:	In this potential attack, an attacker has compromised a private key and generated a signature with it. Let's suppose that the user detects it, and requests the revocation of his certificate c1, indicating time t_0 as the time on which he suspects that the private key was compromised (i.e. invalidityDate, according to [57]). The certification authority (CA) receives the revocation request at time t_1, but does not process it till time t_3. Meanwhile, at time t_2 (t_1 < t_2 < t_3) the CA publishes a new CRL without the revocation information about c1. Therefore, delay t_3 - t_1 prevents the CA from publishing a properly updated CRL at time t_2. A verifier that is validating certificate c1 at a time later than t_0 but before t_2, and following current standards recommendations, waits the grace period before concluding about the validity or invalidity of such certificate. Because next CRL is published at time t_2, that is the one used for the certificate status validation, reaching the conclusion that certificate c1 is valid.
Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result

Method:	D2-CAT6: Influence on certificate verification result → D2-CAT6.2: Alteration of certificate status verification → D2-CAT6.2.1: Grace or cautionary period bypassing → D2-CAT6.2.1.3: Exploit delay in CA's revocation request processing
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.6: CA
Countermeasures:	Use updated revocation information, possibly by accessing an Online Certificate Status Protocol (OCSP) service.

Name:	Low-level LDAP injection techniques to avoid detection of revoked certificate
Source:	This thesis.
Description:	An attacker that is capable of modifying the status validation request made by the verifier will prevent him from checking the actual status of the certificate. Therefore, although the certificate was revoked by the user due to a key compromise, the attacker will make the verifier conclude that the signature is valid. LDAP injection techniques [7] can be used to modify the LDAP query that contains the certificate subject Distinguished Name, making the LDAP server search for a different or nonexistent object. Contrary to classical LDAP injection techniques, where the LDAP query is altered by the attacker due to the malicious input entered from a client application (e.g. Web browser), in this attack the query must be modified at a lower level, for example, before the SVA sends the query to the LDAP server, and once it has been composed.
Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result
Method:	D2-CAT6: Influence on certificate verification result → D2-CAT6.2: Alteration of certificate status verification → D2-CAT6.2.2: Modification of certificate status verification request → D2-CAT6.2.2.2: Modification of LDAP-based request
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.5: SVA
Countermeasures:	Protect queries and responses from integrity attacks (e.g. LDAP-s), and check whether the given response's search criteria matches with the desired one.

Name:	Modification of the OCSP response to avoid detection of revoked certificate (1)
Source:	This thesis.
Description:	This potential attacks requires the attacker to be capable of modifying the OCSP response and subvert the OCSP-response signature verification mechanism in order to prevent the verifier from detecting the violation of the signature integrity. Therefore, it is assumed that the OCSP response has been signed by the OCSP server.

	In this particular attack, the attacker modifies the field *OCSPResponse.responseBytes.response.tbsResponseData.responses[i] .certStatus*, setting its value to 'good'. To subvert the signature verification mechanism, the attacker should apply mechanisms covered by *D2-CAT7: Influence on signature verification result* category, what would fall into a secondary attack not considered herein for classification.
Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result
Method:	D2-CAT6: Influence on certificate verification result → D2-CAT6.2: Alteration of certificate status verification → D2-CAT6.2.3: Modification of certificate status verification response
Target(s):	D3-CAT5: Information → D3-CAT5.2: Protocol message
Countermeasures:	-

Name:	Modification of the OCSP response to avoid detection of revoked certificate (2)
Source:	This thesis.
Description:	This potential attacks requires the attacker to be capable of modifying the OCSP response, signing it with a certificate of his own, and subvert the mechanisms that verify the certification chain. In particular, the attack would cover the modification of the field *OCSPResponse.responseBytes.response.tbsResponseData.responses[i] .certStatus*, setting its value to 'good'. The operations of signing the modified OCSP response with a certificate of his own, and injecting as trust point such certificate, fall into a secondary attack, covered by *D2-CAT6: Influence on certificate verification result → D2-CAT6.3: Untrusted trust anchor/trust point addition* subcategory.
Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result
Method:	D2-CAT6: Influence on certificate verification result → D2-CAT6.2: Alteration of certificate status verification → D2-CAT6.2.3: Modification of certificate status verification response
Target(s):	D3-CAT5: Information → D3-CAT5.2: Protocol message
Countermeasures:	-

Name:	Modification of time-stamp to avoid detection of revoked certificate
Source:	This thesis.
Description:	This potential attack requires the attacker to be able to modify the time-stamp of the signature without detection. Possible mechanisms that can be used further to avoid such detection include subcategories under *D2-CAT6: Influence on certificate verification result* category and *D2-CAT7: Influence on signature verification result* category.
Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result

Method:	D2-CAT6: Influence on certificate verification result → D2-CAT6.2: Alteration of certificate status verification → D2-CAT6.2.4: Alteration of time reference verification → D2-CAT6.2.4.1: Modification of time-stamp
Target(s):	D3-CAT5: Information → D3-CAT5.3: Cryptographic material → D3-CAT5.3.2: Time-stamp
Countermeasures:	If the verifier receives a signature a long time after the time indicated by the time-stamp, then the attack described herein could have been applied. A security policy should indicate whether the signature should be considered as invalid or not, depending on such elapsed time.

Name:	Document masquerading during a document authorization chain
Source:	This thesis.
Description:	This potential attack violates the What-Is-Presented-Is-What-Is-Signed (WIPIWIS) principle. In a situation where a signer has to authorize or approve a signed document authored by another (e.g by countersigning a signature) but after its verification, it might be of interest to the attacker to alter the visualization of the signed document in order to show the intended one. As a result, the authorization would be produced, but over the fraudulent document. In this attack, it is assumed that the attacker has been able to obtain a signature on behalf of the purported signer over a fraudulent document, and that the attacker possesses the intended document as well. Afterwards, the attacker sends to the SVA the pair fraudulent document-signature, what is correctly verified, but makes the SVA show the intended document to the second signer.
Goal:	D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.1: Presentation manipulation → D2-CAT7.1.1: DTBV masquerading → D2-CAT7.1.1.1: Document masquerading
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.5: SVA
Countermeasures:	-

Name:	Showing a different signer during the signature verification
Source:	This thesis.
Description:	This potential attack violates the What-Is-Presented-Is-What-Is-Signed (WIPIWIS) principle, regarding the signed attribute signing-certificate, as defined by Advanced Electronic Signature Formats (AdES) [74, 75]. In this attack, the attacker makes the SVA show a signer different that the actual one. This attack could be launched once the SVA has read the information contained in the certificate signed as attribute (signing-certificate attribute), and possibly by modifying regions of the visualization area of the application (see Cut and paste attacks with Java [150]).

Goal:	D1-CAT4: Make the signed document be attributed to a user different than the actual signer
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.1: Presentation manipulation → D2-CAT7.1.1: DTBV masquerading → D2-CAT7.1.1.2: Attribute masquerading
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.5: SVA
Countermeasures:	-

Name:	Injection of different signature-signed data pair during verification
Source:	This thesis.
Description:	In this potential attack, it is assumed that the attacker possesses a document signed by the signer and the corresponding signature, but different to the signed document and signature that is to be verified. Therefore, the attacker replaces the information during the verification process by injecting into the SVA the former pair of signed document-signature. For example, if two versions of a draft document have been signed by the author, but he only wanted to distribute the newest one for approval, the attacker might want to replace the draft and corresponding signature by the oldest pair.
Goal:	D1-CAT5: Make the Data To Be Verified (DTBV) be shown with chosen content
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.3: Alteration of verification process → D2-CAT7.3.1: Injection of signature-signed data pair
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.5: SVA
Countermeasures:	-

Name:	Modification of cryptographic verification result
Source:	This thesis.
Description:	In this potential attack, if the attacker had access to the routine of the cryptographic verification, then the attacker would be able to make a signature be verified as valid when the integrity was broken.
Goal:	D1-CAT6: Make the signature validity verification conclude with an opposite result
Method:	D2-CAT7: Influence on signature verification result → D2-CAT7.3: Alteration of verification process → D2-CAT7.3.2: Alteration of cryptographic verification result
Target(s):	D3-CAT2: Software → D3-CAT2.1: Application → D3-CAT2.1.2: Related application → D3-CAT2.1.2.5: SVA
Countermeasures:	-

Appendix C

Extended Signature Policy Validation Algorithm

This Appendix contains the pseudo-code of the validation algorithm described in Chapter 8. Firstly, Section C.1 provides the graphical representation used to explain particular operations carried out during the execution of the algorithm. Secondly, the validation algorithm is explained in Section C.2.

C.1 Graphical Representation of a Tree

The notation given in Chapter 1 is used in the graphical representation of a tree, explained herein.

Both the tree of signatures (TSi) and the set of signatures (SSi) processed by the validation algorithm are represented as trees composed of circles and arrows. In the TSi, the circle represents a node by using the node's signer identifier (it is assumed that each node has been given a unique identifier). Near each circle, the acceptable signature policies and the allowed commitment types are indicated among brackets. An absolute dependence (ATS), if present, is represented below the commitment types. Relative and timing dependences (RTS) are represented by an arrow with a discontinuous line. The node from which the arrow starts is the one that has the dependence with the node to which the arrow head points. Thereby, if there is an arrow with discontinuous line from node 3 to node 4, it means that node 3 has a relative dependence (has to be generated after) with node 4. If there is a maximum delta defined for that dependence, then the value is assigned as a label to the arrow. Finally, countersignatures are nodes that are connected by arrows, but in a lower level of the tree (e.g. A node in level two connected to a node in level one means that the former is the countersignature of the latter).

In the SSi, the circle represents a signature (*ds:Signature* element) according to [233] and [75], but with the information simplified. Each signature (circle) has the subject distinguished name inside, while the signature policy used and the commitment type made appear near it. We assume that every signer has selected the same extended signature policy reference, and thus it is not shown. The signing time is represented as a time mark in yyyymmddhhmmss format. The countersignatures are represented in the same way as in the TSi.

Figure C.1 is a simple example of a graphical representation of both a TSi and a SSi, where the node *PS1* of TSi has a RTS on node *PS2*, which has an ATS. On the right side of the Figure, both signatures of the SSi include the signing time.

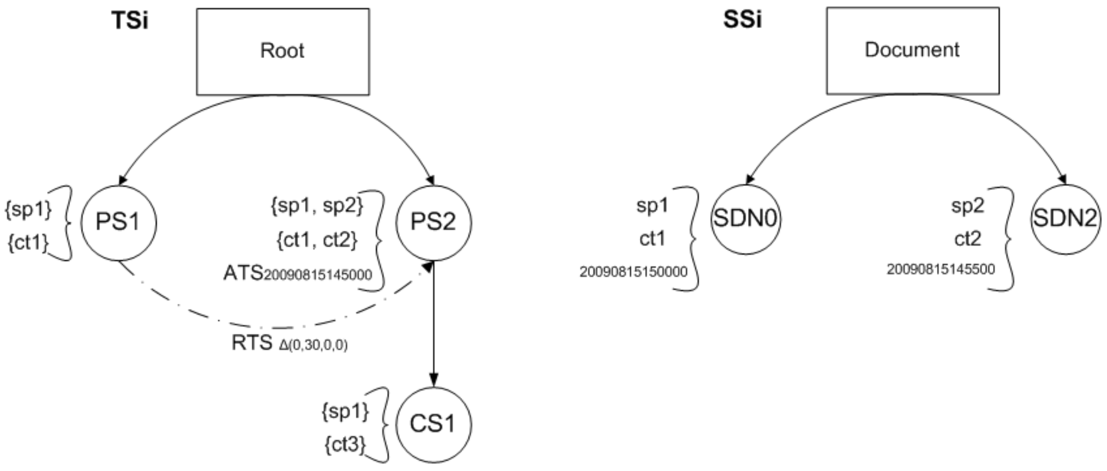

Figure C.1: An example of a graphical representation of a tree.

C.2 Validation Algorithm

The algorithm is split in several routines, which generally follow a recursive design. Prior to the execution of this algorithm, the extended electronic signature policy (ext-SP) must be processed and its information stored in adequate data structures for further analysis.

The detailed data structures, programming language, etc. to be used is out of scope of the Appendix. However, and as will be seen further, references or pointers between certain structures are proposed to accelerate and ease the process. Sometimes, double-linked elements are used. For instance, the algorithm is able to retrieve both the children and parent nodes of a particular node in a TSi. That is, the data structure that represents a node in a TSi (*Signature* ASN.1 type) contains a pointer to each child node but also a pointer to the parent node. These pointers allow the algorithm to explore the TSi both in a forward and a backward manner. An implementation of the

Document Object Model (DOM) of the W3C [65] would achieve that, like Xalan and Xerces, that implement the *org.w3c.dom* Java interface.

The algorithm uses a global variable named *matchingList*, which is accessible from any routine and contains the definite and non-definite signer identifiers assignments between SSi signatures (*subjectDN* field) and TSi nodes (*signer* field).

Algorithm 1 ValidationAlgorithm (Tree TSi, Tree SSi)

```
1:  while ∃ TSi ∈ extSP do
2:    if distribution(TSi[0]) == distribution(SSi[0]) then
3:      for each PrimarySignature ps ∈ SSi[0] do
4:        candidateNodes ← PruneByDimension(ps,TSi[0])
5:        if Explore(ps, candidateNodes) == null then
6:          skip this TSi
7:        end if
8:      end for
9:      if !skip AND Refine(TSi,SSi) then
10:       return true
11:     end if
12:   end if
13: end while
14: return false
```

As can be seen in line 1 of *ValidationAlgorithm* routine, the algorithm processes every TSi until one of them is satisfied (SSi satisfies the ext-SP), or all of them have been analyzed (SSi does not satisfy the ext-SP). As the very first step (2), the algorithm checks the distribution (Distribution-based pruning) of the TSi and SSi root nodes. If there is no correspondence, then it is obvious that this TSi cannot be satisfied by the SSi. Otherwise, the algorithm starts exploring the TSi with each Primary Signature. Before invoking the *Explore* routine (line 5), the algorithm applies the Dimension-based pruning to the nodes in first level of the TSi, taking into account the dimension of the signature being analyzed 4. The resultant list of nodes are the candidate nodes for that signature. See Section 8.2.2.2 of Chapter 8 for information about the functional behavior of these two prunings, and Section C.2.3 of the current Appendix for the corresponding routines.

As explained in Section 8.2.2.3 of Chapter 8, a refinement stage is necessary in order to detect potential deadlocks and unvisited nodes, and evaluate timing and sequence dependences. These tasks are performed by the *Refine* routine (9) in line 9, if and only if exploration of every Primary Signature of SSi has succeed.

C.2.1 Exploring the TSi

The *Explore* routine (2) tries to match the indicated signature with the candidate nodes, progressing along the tree in a recursive manner. The routine discards those nodes that do not have the same distribution as the received signature (lines 1-5). Afterwards, the remaining nodes are evaluated according to the matching criteria (see Routine 3). If the number of matched nodes is zero, then a deadlock occurs (line 10). Each matched node is expanded (line 16) in the sense that its children nodes will be used as candidate nodes for the CounterSignatures (line 19). Children candidate nodes are grouped by branch, in the sense that the child nodes of one candidate node are separated from child nodes of another candidate node. This is necessary in order to correctly apply the Dimension-based pruning before recursively invoking the *Explore* routine 22.

The exploration process follows a DFS strategy. As can be seen in line 35, once every CounterSignature has been processed, the routine returns the list of parent nodes that derived in a matching. Parent nodes not included in the list are discarded as matched nodes for the parent signature during backtracking (line 29). Thereby, the Path-based pruning explained in Section 8.2.2.2 of Chapter 8 is enforced. But before that, the Signer-based pruning is applied as specified in Section 8.2.2.2 (line 28). If every node is discarded by the pruning, a deadlock occurs (line 30).

In order to improve the routine performance, the set of candidate nodes for subsequent CounterSignatures is refined in each iteration by deleting from the list of candidate nodes those that have been pruned (line 33).

C.2.2 Signature Matching

The *Matching* routine (3) evaluates which nodes among the indicated candidates can be matched with the signature. For that purpose, the routine firstly decides if it is the first matching process for the signature's subject distinguished name (*subjectDN*) (line 2). In that case, and if a matching is produced, the node's *signer* field is associated with the *subjectDN*, as shown in line 12. Otherwise, in subsequent executions, this routine will conclude a matching providing that the node's *signer* identifier is already associated with the *subjectDN* (line 15).

When a node is matched, it's visiting counter is increased in one (lines 9 and 16) and a cross pointer is created between the node and the signature (lines 10, 11 and 17, 18). If only one node has been matched with the signature, then a definite assignment is produced (line 23).

When some sort of pruning is performed (see Section C.2.3), or other circumstances arise (e.g. a dependence is not fulfilled – see Section C.2.4.3), the matching between a

Routine 2 Explore (Signature s, List<Node> candidateNodes)

```
1: for each Node n ∈ candidateNodes do
2:    if distribution(s.counterSignatures) != distribution(n.childrenNodes) then
3:       delete n from candidateNodes
4:    end if
5: end for
6: if $|candidateNodes|$ == 0 then
7:    return null
8: end if
9: s.matchedNodes = Matching(s, candidateNodes)
10: if $|s.matchedNodes|$ == 0 then
11:    return null
12: end if
13: childrenCandidateNodes = new List()
14: i = 0
15: for each Node n ∈ s.matchedNodes do
16:    childrenCandidateNodes.add(i,n.childrenNodes)
17:    i++
18: end for
19: for each CounterSignature cs ∈ Signature s do
20:    finalCadidates = new List()
21:    for each $i = 0$ to childrenCandidateNodes.length do
22:       finalCadidates.add(PruneByDimension(cs, childrenCandidateNodes[i]))
23:    end for
24:    activeNodes = Explore(cs, finalCadidates)
25:    if activeNodes == null then
26:       return null
27:    end if
28:    PruneBySigner(s)
29:    PruneByPath(s, activeNodes)
30:    if $|s.matchedNodes|$ == 0 then
31:       return null
32:    end if
33:    delete nodes from childrenCandidateNodes which parent ∉ s.matchedNodes
34: end for
35: return list of parents of s.matchedNodes
```

signature and a node has to be undone. In that case, the *UndoMatching* routine (4) is invoked.

In the *UndoMatching* routine, the corresponding node's visiting counter is decreased in one and the cross pointer deleted. Moreover, the association between the signature

Routine 3 Matching (Signature s, List<Node> candidateNodes)

```
1: matchedNodes = new List()
2: if matchingList[s.subjectDN] = null then
3:    insert ← true
4: end if
5: for each Node n ∈ candidateNodes do
6:    if s.signaturePolicy ∈ n.acceptableSignPolicies AND
      s.commitmentType ∈ n.allowedCommitmentTypes then
7:       if insert then
8:          if n.signer is not definitely assigned then
9:             n.visited++
10:            matchedNodes.add(n)
11:            n.matchedSignatures.add(s)
12:            matchingList.add(s.subjectDN, n.signer)
13:         end if
14:      else
15:         if n.signer ∈ matchingList[s.subjectDN] then
16:            n.visited++
17:            matchedNodes.add(n)
18:            n.matchedSignatures.add(s)
19:         end if
20:      end if
21:   end if
22: end for
23: if $|matchedNodes| = 1$ then
24:    establish n.signer and s.subjectDN as definitely assigned to each other
25: end if
26: return matchedNodes
```

subjectDN and the node signer identifier is deleted if no other node with that signer identifier remains matched with the signature after the matching undo (lines 4-11).

C.2.3 Pruning Methods

In order to improve the algorithm performance, four types of pruning are implemented: the Signer-based pruning, in the *PruneBySigner* routine C.2.3.1, the Path-based pruning, in the *PruneByPath* routine C.2.3.2, the Distribution-based pruning, in the *Distribution* routine C.2.3.3, and the Dimension-based pruning, in the *PruneByDimension* routine C.2.3.4.

Routine 4 UndoMatching (Signature s, Node n)

```
1: n.visited−−
2: delete n from s.matchedNodes
3: delete s from n.matchedSignatures
4: for each Node aux $\in$ s.matchedNodes do
5:    if aux.signer == n.signer then
6:       found $\leftarrow$ true
7:    end if
8: end for
9: if !found then
10:   matchingList.delete(s.subjectDN, n.signer)
11: end if
```

C.2.3.1 Signer-based pruning

This pruning is enforced by the *PruneBySigner* routine (5). In lines 1-5, each node which *signer* field has been definitely assigned to a *subjectDN* different than the signature's one is deleted from the signature's matched nodes.

Each time a matching has to be undone, *UndoMatching* routine is invoked (4).

If only a single node remains matched after the pruning is done, then a definite signer assignment is produced, as indicated in line 6.

Routine 5 PruneBySigner (Signature s)

```
1: for each Node $n \in$ s.matchedNodes do
2:    if n.signer is definitely assigned AND
      matchingList[n.signer] != s.subjectDN then
3:       $UndoMatching$(s,n)
4:    end if
5: end for
6: if $|s.matchedNodes| = 1$ then
7:    establish n.signer and s.subjectDN as definitely assigned to each other
8: end if
```

C.2.3.2 Path-based pruning

The path-based pruning of the algorithm is coded in the *PruneByPath* routine (6). Each node not included in the set of active nodes is deleted from the signature's matched nodes.

Matching undoes and definite signer assignments are treated in the same way as in signer-based pruning routine above.

Routine 6 PruneByPath (Signature s, List<Node> activeNodes)

```
1: for each Node $n \in$ s.matchedNodes do
2:     if $n \notin$ activeNodes then
3:         $UndoMatching$(s,n)
4:     end if
5: end for
6: if $|s.matchedNodes| = 1$ then
7:     establish n.signer and s.subjectDN as definitely assigned to each other
8: end if
```

C.2.3.3 Distribution-based pruning

This pruning is enforced by the *Distribution* routine (7) plus the comparison operation that must be carried out between two distributions when the pruning needs to be done (e.g. line 2 in the *Explore* routine). Therefore, this routine calculates the distribution for a particular collection of nodes (actually, children of the node which distribution has to be calculated).

For that purpose, the routine calculates, for each node, the number of nodes in the collection that have the same identifier (signer identifier if the node belongs to a TSi and *subjectDN* if it belongs to a SSi) (lines 2-16). Each time a node has been analyzed, the counter for the dimension found is updated (lines 9-14).

Finally, the counter for each dimension is normalized due to the counter mode followed. For instance, if the collection of nodes is $\{n_1, n_2, n_2, n_3, n_3, n_3\}$, then, once the loop of line 2 finishes, the distribution will be $\{1 = 1, 2 = 4, 3 = 9\}$, what is wrong. It is because each occurrence is counted as many times as the number of nodes that match. Lines 17-19 correspond to the normalization method applied, which corrects the previous inaccuracy.

Note that in line 15 the identifier of the node is added to a list of identifiers for that dimension. This information will be useful when applying the Dimension-based pruning C.2.3.4.

C.2.3.4 Dimension-based pruning

Finally, this pruning is enforced by the *PruneByDimension* routine (8). Before invoking this routine, the distributions for both the collection of TSi nodes and SSi signatures have to be loaded by invoking the *Distribution* routine (7). The *signaturesInformation* and *nodesInformation* structures correspond to the *information* structure used in the *Distribution* routine.

This routine obtains the dimension of the received signature, and later returns a subset of the received collection of nodes with the nodes which dimension coincides.

Routine 7 Distribution (List<Node> nodesCollection)

```
distribution = new Map()
for each Node n ∈ nodesCollection do
  dimension = 0
  for each Node n' ∈ nodesCollection do
    if n.identifier == n'.identifier then
      dimension++
    end if
  end for
  counter = distribution[dimension]
  if counter == null then
    distribution[dimension] = 1
  else
    distribution[dimension] = counter + 1
  end if
  information[dimension].add(node.identifier)
end for
for each dimension d ∈ distribution do
  distribution[d] = distribution[d]/d
end for
```

Routine 8 PruneByDimension (Signature s, List<Node> nodes)

```
nodesToReturn ← new List()
for each dimension d ∈ signaturesInformation do
  if signaturesInformation[d] contains s.subjectDN then
    for each Node n ∈ nodes do
      if nodesInformation[d] contains n.signer then
        nodesToReturn.add(n)
      end if
    end for
    return nodesToReturn
  end if
end for
```

C.2.4 Refinement Stage

The *Refine* routine (9) consists of three phases, named *RefinementPhaseOne* (deadlocks detection) C.2.4.1, *RefinementPhaseTwo* (unvisited nodes detection) C.2.4.2 and *RefinementPhaseThree* (timing and sequence constraints evaluation) C.2.4.3, further explained. Phases one and two must be executed again after phase three (lines 14-23) because *RefinementPhaseThree* can undo matchings between nodes and signatures.

Routine 9 Refine (Tree TSi, Tree SSi)

```
1: repeat
2:   for each PrimarySignature ps $\in$ SSi[0] do
3:     if $RefinementPhaseOne$(ps) = null then
4:       return false
5:     end if
6:   end for
7: until no change
8: if !$RefinementPhaseTwo$(TSi) then
9:   return false
10: end if
11: if !$RefinementPhaseThree$(TSi) then
12:   return false
13: end if
14: repeat
15:   for each PrimarySignature ps $\in$ SSi[0] do
16:     if $RefinementPhaseOne$(ps) = null then
17:       return false
18:     end if
19:   end for
20: until no change
21: if !$RefinementPhaseTwo$(TSi) then
22:   return false
23: end if
24: return true
```

C.2.4.1 Detecting potential deadlocks

During the first phase of the refinement, the *RefinementPhaseOne* routine (10) carries out a path-based and signer-based prunings both in a top-down and bottom-up approaches.

During the exploration process (see Routine 2), the prunings are mainly applied in a bottom-up approach. The top-down approach is followed when exploring sibling nodes, from left to right. This strategy makes a pruning applied to a certain node not having effect in a left-hand side sibling node. As a result, a signature can maintain matchings with nodes which parent nodes are further pruned for the parent signature, or a signature can maintain a matching with a node which *signer* identifier is later definitely assigned to a different *subjectDN*.

To resolve this issue, the *RefinementPhaseOne* routine deletes the nodes belonging to the matched nodes which parent nodes are not matched with the parent signature (lines 3-7). Afterwards. the signer-based and path-based prunings are performed in lines 8 and 9, respectively. Obviously, if no matched node remains after the pruning, a deadlock occurs.

The SSi is explored in a DFS strategy until the leaf signatures are reached. From that moment onwards, the routine returns, in each executed recursive iteration, the list of parent nodes corresponding to the matched nodes (line 25), and the obtained active nodes are used for the path-based pruning (line 20), after having applied the signer-based pruning (line 19). This behavior implements the bottom-up pruning approach, like in the *Explore* routine.

As can be seen in *Refine* routine (9), the *RefinementPhaseOne* routine is executed until no change is produced, that is, a stable version of the solution is obtained or a deadlock occurs.

Routine 10 RefinementPhaseOne (Signature s)

```
 1: activeNodes = s.matchedNodes
 2: if s→parent $\neq$ null then
 3:    for each Node $n \in$ activeNodes do
 4:       if n→parent $\notin$ s→parent.matchedNodes then
 5:          delete n from activeNodes
 6:       end if
 7:    end for
 8:    $PruneBySigner$(s)
 9:    $PruneByPath$(s, activeNodes)
10:    if $|s.matchedNodes|$ == 0 then
11:       return null
12:    end if
13: end if
14: for each CounterSignature cs $\in$ signature s do
15:    activeNodes = $RefinementPhaseOne$(cs)
16:    if activeNodes == null then
17:       return null
18:    end if
19:    $PruneBySigner$(s)
20:    $PruneByPath$(s, activeNodes)
21:    if $|s.matchedNodes|$ == 0 then
22:       return null
23:    end if
24: end for
25: return list of parents of matchedNodes
```

C.2.4.2 Detecting unvisited nodes

The *RefinementPhaseTwo* routine (11) is quite simple. It merely explores the tree TSi following a BFS strategy, by using a FIFO (First In First Out) queue. If an unvisited node is found (line 6), then the refinement fails (and so does the validation).

Routine 11 RefinementPhaseTwo (Tree TSi)

```
1: for each Node n ∈ TSi[0] do
2:    Queue ← node
3: end for
4: while Queue has elements do
5:    aux ← Queue
6:    if aux.visited == 0 then
7:       return false
8:    end if
      Queue ← aux.childrenNodes
9: end while
```

C.2.4.3 Evaluating the timing and sequence dependencies

The timing and sequence dependences are evaluated during the phase three, represented in the *RefinementPhaseThree* routine (12). In this phase, the TSi is explored in a BFS approach, as in phase two, and each node's timing and sequence constraints are evaluated. A node's dependence can be either absolute (line 8) or relative (line 12).

Absolute timing and sequence evaluation

Absolute dependences are evaluated in the *EvalAbsTimeSeq* routine (13). A matching is undone for each signature matched with the received node that does not fulfill the constraint. Note that when a matching is undone (line 3), it has to be expanded through the TSi tree by means of the *UndoMatchingProgressAtOrigin* routine (17), further explained.

Relative timing and sequence evaluation

Relative dependences are evaluated in the *EvalRelTimeSeq* routine (14). In this case, the evaluation is a bit more complex. A node can have several relative dependences with other nodes. As each node can be matched with more than one signature, many possible combinations may arise.

Each relative dependence is separately analyzed (line 2). The node with which there is a dependence is searched in the TSi by using the path of node's identifiers (line 3 - Routine 15). From line 4 to line 17, each combination of signatures is evaluated. Each satisfying combination is added to the set of dependence solutions (line 7), and it is said to be a solution pair. A solution pair consists of an origin signature and its corresponding matched node, and a referenced signature and its corresponding matched node. The origin node is the node that has the relative dependence on the referenced one, and the origin and referenced signatures are signatures which signing time properties comply with the constraint.

If no solution pair is generated for a certain signature of the node being evaluated, the matching is undone (line 10) by calling the *UndoMatching* routine (4). Furthermore,

Routine 12 RefinementPhaseThree (Tree TSi)

```
1:  for each Node n ∈ TSi[0] do
2:     Queue ← node
3:  end for
4:  while Queue has elements do
5:     aux ← Queue
6:     dependence ← aux.timingAndSequence
7:     if dependence is absolute then
8:        if !EvalAbsTimeSeq(TSi, aux) then
9:           return false
10:       end if
11:    else
12:       if !EvalRelTimeSeq(TSi, aux) then
13:          return false
14:       end if
15:    end if
       Queue ← aux.childrenNodes
16: end while
17: return true
```

Routine 13 EvalAbsTimeSeq (Tree TSi, Node n)

```
1:  for each Signature $s$ ∈ n.matchedSignatures do
2:     if $s$ DOES NOT satisfy dependence then
3:        UndoMatching(s, n)
4:        if $|s.matchedNodes|$ == 0 OR $|n.matchedSignatures|$ == 0 then
5:           return false
6:        end if
7:        UndoMatchingProgressAtOrigin(TSi, s, n)
8:     end if
9:  end for
10: return true
```

as this node can be referenced from others' relative dependences, the deletion must be progressive, and thus the *UndoMatchingProgressAtOrigin* routine 17.

If no signature remains matched with the node after a relative dependence evaluation, then the node becomes unvisited, and the refinement fails (and so does the validation) (line 18). Moreover, signatures matched with a node referenced in a dependence, and that are not included in any generated solution pair, are deleted from that node by using a progressive deletion at destination routine (lines 21-31) (see *UndoMatchingProgressAtDestination* routine 19).

In a nutshell, the relative timing and sequence dependences are satisfied if there is at least one signature matched with the node that satisfies every dependence. In a formal way, this requirement is expressed as follows:

$$\begin{aligned} &\exists\ s \in\ n.matchedSignatures\ /\ \exists\ solutionPair(s, s') \\ \forall\ rd \in\ n.dependences,\ &s'\ \in\ rd->refNode.matchedSignatures \end{aligned} \tag{C.1}$$

where

s is a signature matched with node n

rd is a relative dependence existent in node n

s' is a signature matched with node $refNode$ referenced in the dependence rd

When a relative dependence has to be evaluated, the node with which the dependence exists must be searched in TSi, as shown in line 3 of *EvalRelTimeSeq* routine (14). However, the information available in that moment is the path of unique nodes' identifiers from the root node to the referenced node. The *NodeSearch* routine (15) uses the path to find the node by using a BFS strategy focused on a specific level in each iteration (16).

The *ExclusivePathBasedSearch* routine (Routine 16) processes a specific node taking into account a node identifier included in the path (line 1). If the node's identifier match, then the routine explores, in a recursive manner, the children nodes but using the next identifier in the path (lines 5-14). The routine returns "error" if every possibility has been analyzed without finding the corresponding node, "node not found" if the specific evaluated node does not match the particular node identifier, or the node ("goal") if it is found and the path has been completely processed (line 2).

Progressive deletion routines

During the refinement phase three (C.2.4.3), an unmatching between a signature and a node can be produced if some timing and sequence dependence is not fulfilled. Next,

Routine 14 EvalRelTimeSeq (Tree TSi, Node n)

```
1: dependence ← n.timingAndSequence
2: for each relativeDependence $rd$ ∈ dependence do
3:   rNode = $NodeSearch$(TSi, rd.pathToRefSignature))
4:   for each Signature $s$ ∈ n.matchedSignatures do
5:     for each Signature $rSignature$ ∈ rNode.matchedSignatures do
6:       if $s$ and $rSignature$ satisfy dependence then
7:         n.dependences ← add new solutionPair(($s$,$n$),($rSignature$,$rNode$))
8:       end if
9:     end for
10:    if no pair created for Signature $s$ then
11:      $UndoMatching$(s,n)
12:      $UndoMatchingProgressAtOrigin$(TSi, s, n)
13:      if $|s.matchedNodes|$ == 0 OR $|n.matchedSignatures|$ == 0 then
14:        return false
15:      end if
16:    end if
17:  end for
18:  if $|n.matchedSignatures|$ == 0 then
19:    return false
20:  end if
21:  for each Signature $rSignature$ ∈ rNode.matchedSignatures do
22:    for each solutionPair $sp$ ∈ rd do
23:      if $sp \rightarrow referencedSignature == rSignature$ then
24:        keep ← true
25:        skip
26:      end if
27:    end for
28:    if !keep then
29:      $UndoMatchingProgressAtDestination$(TSi, rSignature, rNode)
30:    end if
31:  end for
32: end for
33: return true
```

two progressive deletion routines that maintain the consistence of the mapping between the SSi and the TSi are explained.

Undoing a matching at the origin

When a matching is undone, the whole tree has to be evaluated again in order

Routine 15 NodeSearch (Tree TSi, List<int> pathIds)

```
for each Node n ∈ TSi[0] do
  goal = ExclusivePathBasedSearch(n, pathIds, 0)
  if node found then
    return goal
  end if
end for
return node not found
```

Routine 16 ExclusivePathBasedSearch (Node node, List<int> pathIds, int index)

```
if node.id == pathIds[index] then
  if pathIds.length == index +1 then
    return node
  end if
  for each Node childNode ∈ node.childrenNodes do
    goal = ExclusivePathBasedSearch(childNode, pathIds, i+1)
    if node found then
      return goal
    else if error then
      return error
    else
      do nothing
    end if
  end for
  return error
else
  return node not found
end if
```

to detect which nodes are linked with that particular signature and node by means of a solution pair. That is, which nodes have not only a relative dependence with that node but also reference the unmatched signature in a solution pair. This update process is called a progressive deletion at the origin, and it is implemented by the *UndoMatchingProgressAtOrigin* routine (17). This routine can be invoked after both absolute and relative dependence evaluations.

Next Figure C.2 shows a situation where a progressive deletion at the origin has to be executed because an absolute dependence is not fulfilled.

The Figure represents a fragment of a TSi tree with two nodes (PS1 and PS2). PS1 node has been matched with signatures SDN3 and SDN4, while PS2 node has been matched with signatures SDN0 and SDN1. Let suppose that, during the refinement

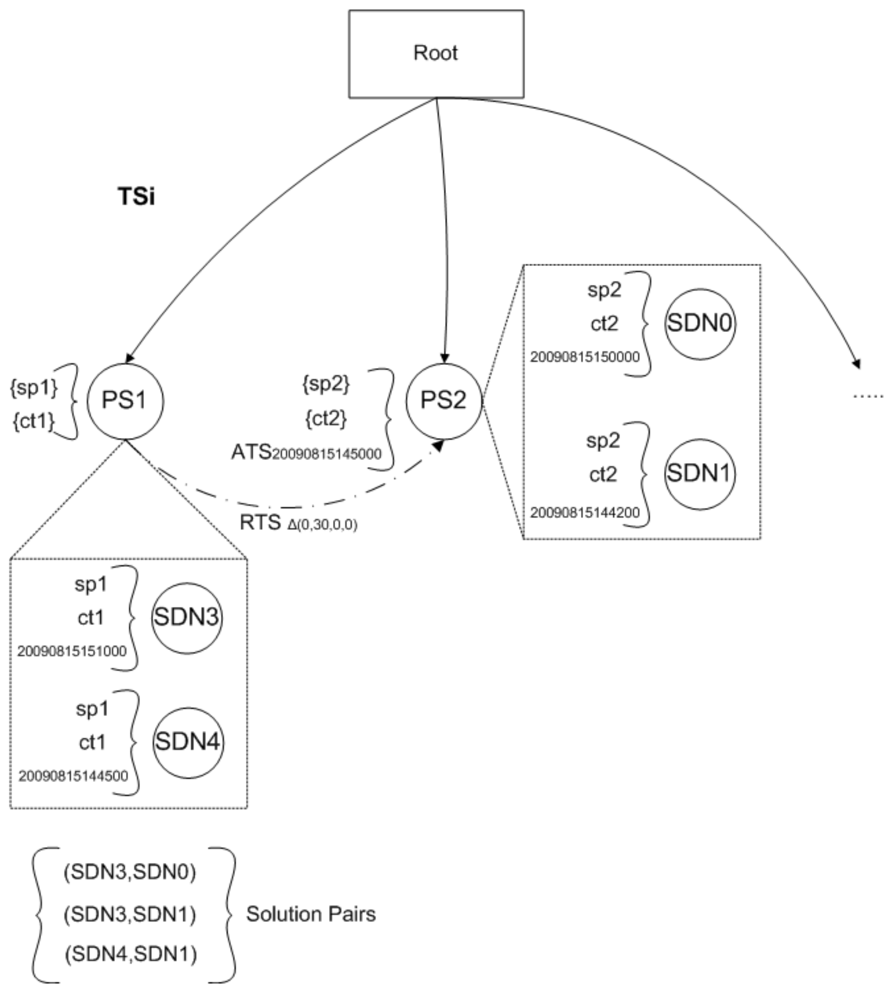
Root
TSi
{sp1} {ct1} PS1
{sp2} {ct2} ATS20090815145000 PS2
RTS Δ(0,30,0,0)
sp2 ct2 20090815150000 SDN0
sp2 ct2 20090815144200 SDN1
sp1 ct1 20090815151000 SDN3
sp1 ct1 20090815144500 SDN4
.....
(SDN3,SDN0)
(SDN3,SDN1)
(SDN4,SDN1)
Solution Pairs

Figure C.2: Undoing a matching at the origin (Absolute dependence not fulfilled).

phase three, PS1 node has been firstly evaluated, being three solution pairs generated. Afterwards, when the algorithm evaluates PS2 node, it detects that only signature SDN0 complies with the absolute timing and sequence dependence – which sets the *NotBefore* constraint only – established. As a result, the matching between node PS2 and signature SDN1 is undone by calling *UndoMatchingProgressAtOrigin* routine (see *EvalAbsTimeSeq* routine 13). Subsequently, the algorithm explores node PS1 again to detect, firstly, if it has a dependence on the PS2 node, what is true. Secondly, the algorithm analyses the solution pairs created to detect any solution pair that references signature SDN1, and delete it. It can be noticed that two of the three solution pairs reference signature SDN1 ($\{SDN3, SDN1\}$ and $\{SDN4, SDN1\}$). The algorithm deletes both of them. As a consequence, no solution pair with signature SDN4 as origin remains, and thus the algorithm undoes the matching between signature SDN4 and PS1 node.

If the TSi tree had more nodes, the algorithm would have to process it in order to find any node with a relative dependence on PS1 node that could reference signature

SDN4 (recursive calling of *UndoMatchingProgressAtOrigin* routine).

On the other hand, Figure C.3 depicts the case of a progressive deletion at origin when a relative dependence is not fulfilled by one of the signatures (SDN1) initially matched with a node (PS2). In this case, the absolute dependence established for PS2 node is fulfilled by both signatures SDN0 and SDN1. However, when evaluating the relative dependence on PS3 node, only signature SDN0 fulfills it, and thus it is the only signature that appears as the origin of the generated solution pairs, with signatures SDN4 and SDN5 of node PS3 as the referenced signatures ($\{SDN0, SDN4\}$ and $\{SDN0, SDN5\}$, respectively).

When the algorithm detects that no solution pair is generated for signature SDN1, the matching between signature SDN1 and PS2 node is undone (operation carried out by the *EvalRelTimeSeq* routine 14, not the *UndoMatchingProgressAtOrigin* one), and the *UndoMatchingProgressAtOrigin* routine is invoked to “clean” the solution pairs of PS1 node (solution pair $\{SDN3, SDN1\}$ is deleted).

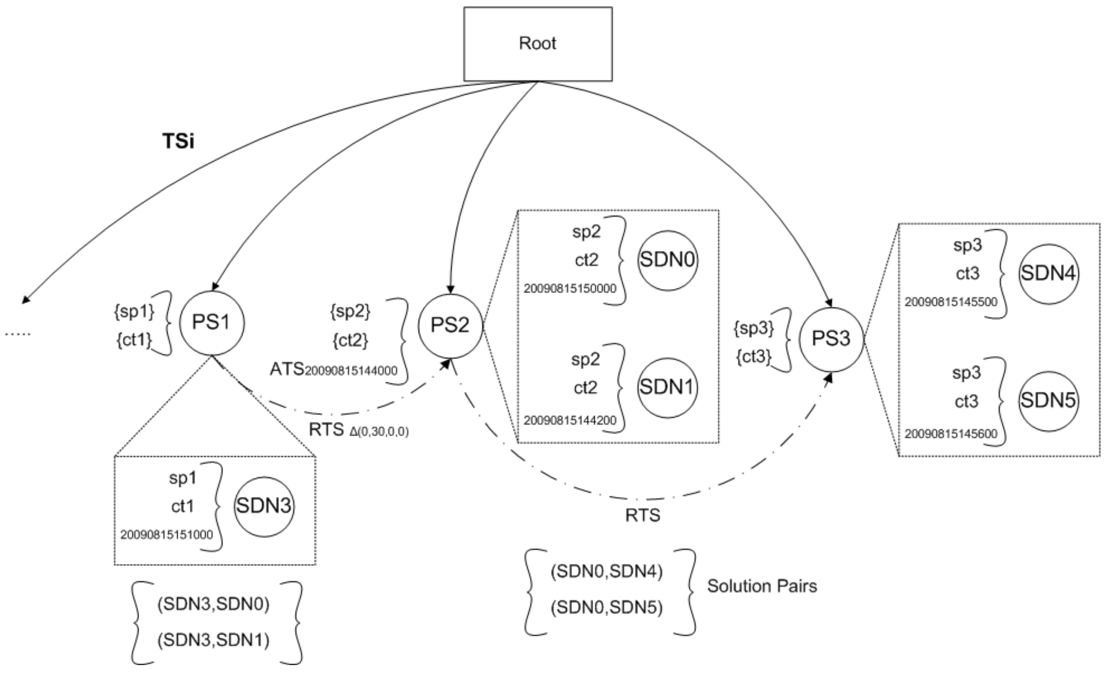

Figure C.3: Undoing a matching at the origin (Relative dependence not fulfilled).

This routine internally invokes the *UndoMatchingProgress* routine (18) to perform the operations herein described.

The *UndoMatchingProgress* routine (18) evaluates if the indicated node has any solution pair for the indicated signature (as a referenced signature). A node is skipped if it has not been evaluated yet from the dependence viewpoint or if the dependence is not relative (line 2). Otherwise, every generated pair that points to the received signature is deleted (lines 3-12).

Routine 17 UndoMatchingProgressAtOrigin (Tree TSi, Signature rSignature, Node rNode)

```
for each Node $n \in$ TSi[0] do
    if $n \neq$ rNode then
        $UndoMatchingProgress$(TSi, n, rSignature, rNode)
    end if
end for
```

Once every solution pair that pointed to the referenced signature has been deleted, it must be checked which signatures matched with this node are not included (as origin signatures) anymore in the solution pairs (in any relative dependence) (lines 13-30). In that cases, a progressive unmatching is carried out (line 24).

The TSi tree is explored following the paths of the children nodes of the particular node (line 31).

Undoing a matching at the destination

When the relative dependence of a node is evaluated, it is possible that some signatures matched with the referenced node are not included in the solution pairs as the referenced signature (see Section *Relative timing and sequence evaluation* C.2.4.3). In those cases, the *UndoMatchingProgressAtDestination* routine (19) is invoked to undo the matching between the referenced node n and the indicated signature s, and to "clean" the solution pairs therein created (if any). Basically, this routine performs four tasks:

1. Node n is analyzed. Every solution pair therein created (obviously, node n must have been processed from a dependence viewpoint – line 3), is checked. Every pair that has, as the origin signature, the indicated one (see line 7), is deleted (line 8).

2. As the signature s is not included in the pairs anymore, it is deleted from the matched signatures of node n (line 13).

3. In order to maintain the consistence of the tree, the *UndoMatchingProgressAtDestination* routine is invoked for every referenced signature that is not included in the solution pairs anymore (lines 17–28).

4. Finally, and as a result of the unmatching between the signature and the node, *UndoMatchingProgessAtOrigin* is invoked (line 29).

Routine 18 UndoMatchingProgress (Tree TSi, Node node, Signature rSignature, Node rNode)

```
1: dependence ← node.timingAndSequence
2: if dependence is relative then
3:   for each dependence $d \in$ node do
4:     if $d \rightarrow referencedNode == rNode$ then
5:       solutionPairs ← d.solutionPairs
6:       for each solutionPair ∈ solutionPairs do
7:         if $solutionPair \rightarrow referencedSignature == rSignature$ then
8:           remove solutionPair
9:         end if
10:      end for
11:    end if
12:  end for
13:  for each Signature $s \in$ node.matchedSignatures do
14:    for each dependence $d \in$ node do
15:      solutionPairs ← d.solutionPairs
16:      for each solutionPair ∈ solutionPairs do
17:        if $solutionPair \rightarrow originSignature == s$ then
18:          maintain this signature $s$ as matched signature for $node$
19:          found ← true
20:        end if
21:      end for
22:    end for
23:    if !found then
24:      $UndoMatching$(s,node)
25:      if $|s.matchedNodes| == 0$ OR $|node.matchedSignatures| == 0$ then
26:        throw Deadlock
27:      end if
28:      $UndoMatchingProgressAtOrigin$(TSi, s, node)
29:    end if
30:  end for
31:  for each childrenNode $cn \in node$ do
32:    $UndoMatchingProgress$(TSi, cn, rSignature, rNode)
33:  end for
34: end if
```

Figure C.4 shows an example where a matching at destination has to be undone. Let suppose that nodes PS4 and PS5 have already been processed. When the algorithm evaluates the relative dependence of PS2 node, it detects that signature SDN6 matched with PS3 node is not contained as a referenced signature in any generated solution pair.

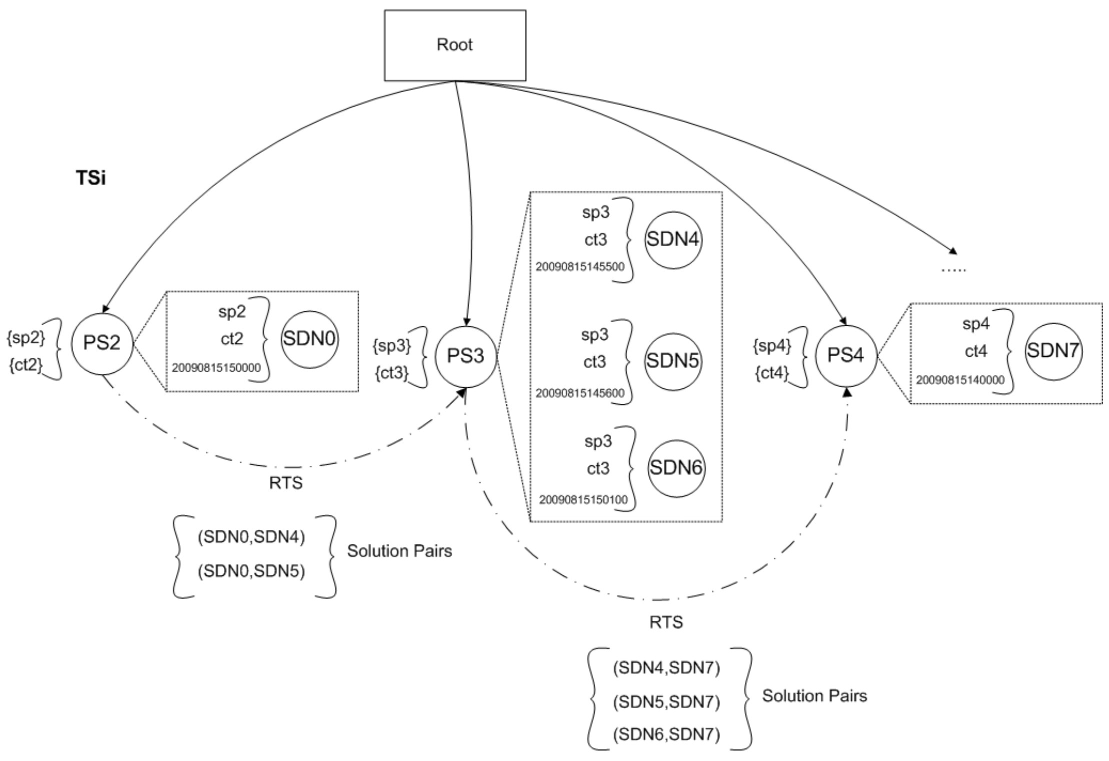

Figure C.4: Undoing a matching at the destination (Relative dependence not fulfilled).

Therefore, the *UndoMatchingProgressAtDestination* routine will review the solution pairs in PS3 in which signature SDN6 appears as origin signature, deleting solution pair $\{SDN6, SDN7\}$ (task one). Consequently, the matching between signature SDN6 and PS3 node, is undone (task two). In this particular case, no referenced signature has been completely deleted from solution pairs of node PS3 (only signature SDN7 was referenced), and thus the *UndoMatchingProgressAtDestination* does not have to be invoked (as said in task 3). However, and as the matching between signature SDN6 and node PS3 has been undone, the *UndoMatchingProgessAtOrigin* routine is invoked (task four), but without producing any effect on the tree.

Routine 19 UndoMatchingProgressAtDestination (Tree TSi, Signature s, Node n)

```
1: dependence ← n.timingAndSequence
2: deletedSolutions ← new List()
3: if dependence is relative then
4:   for each dependence $d \in$ n do
5:     solutionPairs ← d.solutionPairs
6:     for each solutionPair ∈ solutionPairs do
7:       if $solutionPair \rightarrow originSignature == s$ then
8:         remove solutionPair from solutionsPairs
9:         deletedSolutions.add(solutionPair)
10:      end if
11:    end for
12:  end for
13:  $UndoMatching$(s, n)
14:  if $|s.matchedNodes| == 0$ OR $|n.matchedSignatures| == 0$ then
15:    throw Deadlock
16:  end if
17:  for each deletedSolution $ds \in$ deletedSolutions do
18:    for each dependence $d \in$ n do
19:      for each solutionPair $sp \in$ solutionPairs do
20:        if $ds \rightarrow referencedSignature == sp \rightarrow referencedSignature$ then
21:          found ← true
22:        end if
23:      end for
24:    end for
25:    if !found then
26:      $UndoMatchingProgressAtDestination$(TSi, $ds \rightarrow referencedSignature$,
         $ds \rightarrow referencedNode$)
27:    end if
28:  end for
29:  $UndoMatchingProgressAtOrigin$(TSi, s, n)
30: else
31:  $UndoMatching$(s, n)
32:  if $|s.matchedNodes| == 0$ OR $|n.matchedSignatures| == 0$ then
33:    throw Deadlock
34:  end if
35:  $UndoMatchingProgressAtOrigin$(TSi, s, n)
36: end if
```

Appendix D

Test Cases for the Extended Signature Policy Validation Algorithm

This Appendix details the test cases executed to verify the correctness of the validation algorithm designed for the extended signature policy framework. Section D.1 contains the textual representation used to describe the test cases, and by which the space needed for the description of the tests is reduced, and the descriptions homogenized. Sections D.2 and D.3 include the trees of signatures and set of signatures used for the test cases, respectively, while the results of the test cases are given in Section D.4.

D.1 Textual Representation of a Tree

The notation given in Chapter 1 is used in the textual representation of a tree, explained herein.

Nodes in first level of the tree correspond to primary signatures (PS), while nodes located in subsequent levels are countersignatures (CS). Depending on the type of tree being represented (Tree of Signatures, TSi, or Set of Signatures, SSi), the information for each node differs. A node of a TSi is textually represented as follows:

$$\begin{aligned} node &::= [nodeUnit + node] \mid [nodeUnit] \\ nodeUnit &::= id\_type\,(signerId/\,\{sp\}\,/\,\{ct\}\,/TS) \end{aligned}$$

where:

- *id* is the identifier that uniquely identifies this node among the rest of TSis nodes.
- *type* can be is either PS or CS.

- *signerId* is the node's signer identifier (e.g. PS2, CS4). A signer identifier does not depend on the type of node. That is, a signer identifier PS3 is valid for a node in a level different than the first one (reserved for primary signatures).
- *sp* is a list, between brackets, of acceptable signature policies that can be used by the signer. Each signature policy is represented by the particular version used by the signer (e.g. 1.0.1, 1.0.2, etc.). The rest of the URL (`http://jlopez.thesis.uc3m.es/SigPolicy/v`) remains the same for every signature policy, as commented in Section 10.3.1.2, and thus is not included in the representation.
- *ct* is a list, between brackets, of allowed commitment types that can be made by the signer. As introduced in Section 10.3.1.2, the commitment types are established in XAdES-EPES formats with an URI value. In order to reduce the space needed for a particular commitment type, a mapping between the URIs and a short value used for the representation is made, shown in Table D.1.
- *TS* can be either ATS or RTS. It is optional.
- *countersignatures* are represented by including *nodes* inside a node. They are separated by symbol +. They are optional.

A RTS is represented by a reference to the node's unique identifier on which it has the dependence ($RTS\,(id)$). If a maximum delta exists, then it is represented as

Commitment type indication URI	Value
`http://uri.etsi.org/01903/v1.2.2#ProofOfCreation`	2.0.1
`http://uri.etsi.org/01903/v1.2.2#ProofOfApproval`	2.0.2
`http://uri.etsi.org/01903/v1.2.2#ProofOfDelivery`	2.0.3
`http://uri.etsi.org/01903/v1.2.2#ProofOfStorage`	2.0.4
`http://uri.etsi.org/01903/v1.2.2#ProofOfAcknowledgment`	2.0.5
`http://uri.etsi.org/01903/v1.2.2#ProofOfReview`	2.0.6
`http://uri.etsi.org/01903/v1.2.2#ProofOfSecondReview`	2.0.7
`http://uri.etsi.org/01903/v1.2.2#ProofOfThirdReview`	2.0.8
`http://uri.etsi.org/01903/v1.2.2#ProofOfFourthReview`	2.0.9
`http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfOrigin1`	0.1
`http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfReceipt1`	0.2
`http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfOrigin2`	0.3
`http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfReceipt2`	0.4
`http://uri.etsi.org/01903/v1.2.2#NonRepudiationEvidence`	0.5

Table D.1: Mapping between *xades:CommitmentTypeIndication* URIs and values used for the representation of TSis and SSis

$RTS\left(id,\Delta\left(0,0,10,1\right)\right)$.

Next TSi textual representation indicates that the TSi has two Primary Signatures with one CounterSignature each, having the first PS an absolute timing and sequence dependence, and the second one a relative dependence on the first one:

$[1\_PS(PS1/\{sp1,sp2\}/\{ct1,ct2,ct3\}/ATS_{20000131120000})+[2\_CS(CS1/\{sp1\}/\{ct4\})]]+$
$[3\_PS(PS2/\{sp1,sp2\}/\{ct1,ct3\}/RTS(1))+[4\_CS(CS2/\{sp2\}/\{ct3\})]]$

On the other hand, a signature of a SSi is represented with the next information:

$$signature \ ::= \ [signatureUnit + signature] \ | \ [signatureUnit]$$
$$signatureUnit \ ::= \ subjectDN/sp/ct/signingTime$$

where:

- *subjectDN* is the subject distinguished name that identifies the signer.
- *sp* corresponds to the version of the signature policy used by the signer when generating the signature.
- *ct* corresponds to the value mapped to the commitment type (URI) made by the signer (see table D.1).
- *signingTime* is the time at which the signer performed the signature. For clarity purposes, the time value will be represented in yyyymmddhhmmss format (e.g. 19970717000000). This field is optional.
- *countersignatures* are represented by including *signatures* inside a signature. They are separated by symbol +. It is optional.

It is supposed that every signature has referenced the same extended signature policy when generating the signature. As that information will not vary in any test case, it is not included in the textual representation of the TSis and SSis.

The next example shows a SSi textual representation where there are two Primary Signatures in the first level, and one CounterSignature generated over the second one:

$[subjectDN1/sp1/ct1/20000131110000]+[subjectDN2/sp1/ct2/20000131120000+$
$[subjectDN3/sp1/ct3]]$

D.2 Trees of Signatures for the Test Cases

Next, the TSis that have been used as input for the test cases are defined following the textual representation explained in Section D.1.

TSi	**Tree structure**
tsi_00_v0	$[1\_PS(PS1/\{1.0.1, 1.0.2\}/\{2.0.1, 2.0.2\})]$
tsi_00_v1	$[1\_PS(PS1/\{1.0.1, 1.0.2\}/\{2.0.1, 2.0.2\}/ATS_{19970717000000-20170717000000})]$

Table D.2: TSis with one Primary Signature.

TSi	**Tree structure**
tsi_01_v0	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})] + [2\_PS(PS2/\{1.0.1\}/\{2.0.1\})] + [3\_PS(PS3/\{1.0.1\}/\{2.0.1\})]$
tsi_01_v1	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})]+[2\_PS(PS2/\{1.0.1\}/\{2.0.1\}/RTS(1))]+[3\_PS(PS3/\{1.0.1\}/\{2.0.1\}/RTS(2))]$
tsi_01_v2	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})] + [2\_PS(PS2/\{1.0.1\}/\{2.0.1\}/RTS(1, \Delta(0,0,1,0)))]+[3\_PS(PS3/\{1.0.1\}/\{2.0.1\}/RTS(2, \Delta(0,0,1,0)))]$
tsi_01_v3	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})] + [2\_PS(PS1/\{1.0.1\}/\{2.0.1\})] + [3\_PS(PS3/\{1.0.1\}/\{2.0.1\})]$
tsi_01_v4	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})]+[2\_PS(PS1/\{1.0.1\}/\{2.0.1\}/RTS(1))]+[3\_PS(PS3/\{1.0.1\}/\{2.0.1\}/RTS(2))]$
tsi_01_v5	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})] + [2\_PS(PS1/\{1.0.1\}/\{2.0.1\}/RTS(1, \Delta(0,0,1,0)))]+[3\_PS(PS3/\{1.0.1\}/\{2.0.1\}/RTS(2, \Delta(0,0,1,0)))]$
tsi_01_v6	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})] + [2\_PS(PS2/\{1.0.1\}/\{2.0.1\})] + [3\_PS(PS1/\{1.0.1\}/\{2.0.1\})]$
tsi_01_v7	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})]+[2\_PS(PS2/\{1.0.1\}/\{2.0.1\}/RTS(1))]+[3\_PS(PS1/\{1.0.1\}/\{2.0.1\}/RTS(2))]$
tsi_01_v8	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})] + [2\_PS(PS2/\{1.0.1\}/\{2.0.1\}/RTS(1, \Delta(0,0,1,0)))]+[3\_PS(PS1/\{1.0.1\}/\{2.0.1\}/RTS(2, \Delta(0,0,1,0)))]$
tsi_01_v9	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})] + [2\_PS(PS1/\{1.0.1\}/\{2.0.1\})] + [3\_PS(PS1/\{1.0.1\}/\{2.0.1\})]$
tsi_01_v10	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})]+[2\_PS(PS1/\{1.0.1\}/\{2.0.1\}/RTS(1))]+[3\_PS(PS1/\{1.0.1\}/\{2.0.1\}/RTS(2))]$
tsi_01_v11	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})] + [2\_PS(PS1/\{1.0.1\}/\{2.0.1\}/RTS(1, \Delta(0,0,1,0)))]+[3\_PS(PS1/\{1.0.1\}/\{2.0.1\}/RTS(2, \Delta(0,0,1,0)))]$
tsi_01_v12	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})] + [2\_PS(PS2/\{1.0.2\}/\{2.0.2\})] + [3\_PS(PS3/\{1.0.3\}/\{2.0.3\})]$
tsi_01_v13	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})]+[2\_PS(PS2/\{1.0.2\}/\{2.0.2\}/RTS(1))]+[3\_PS(PS3/\{1.0.3\}/\{2.0.3\}/RTS(2))]$
tsi_01_v13b	$[3\_PS(PS3/\{1.0.2\}/\{2.0.2\}/RTS(2))] + [2\_PS(PS2/\{1.0.1\}/\{2.0.1\}/RTS(1))] + [1\_PS(PS1/\{1.0.1\}/\{2.0.1\}/RTS(2))]$
tsi_01_v14	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})] + [2\_PS(PS2/\{1.0.2\}/\{2.0.2\}/RTS(1, \Delta(0,0,1,0)))]+[3\_PS(PS3/\{1.0.3\}/\{2.0.3\}/RTS(2, \Delta(0,0,1,0)))]$
tsi_01_v15	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})] + [2\_PS(PS1/\{1.0.2\}/\{2.0.2\}/RTS(1, \Delta(0,0,1,0)))]+[3\_PS(PS1/\{1.0.3\}/\{2.0.3\}/RTS(2, \Delta(0,0,1,0)))]$
tsi_01_v16	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})] + [2\_PS(PS1/\{1.0.1\}/\{2.0.1\}/RTS(1, \Delta(0,0,1,0)))]+[3\_PS(PS1/\{1.0.2\}/\{2.0.2\}/RTS(2, \Delta(0,0,1,0)))]$

tsi_01_v17	$[3\_PS(PS3/\{1.0.3\}/\{2.0.3\}/RTS(2,\Delta(0,0,5,0)))] + [2\_PS(PS2/\{1.0.1\}/\{2.0.1\}/ \quad ATS_{20090718101000-20090718103000})] + [1\_PS(PS1/\{1.0.1\}/\{2.0.1\}/ ATS_{20090718100000-20090718101500})]$

Table D.3: TSis with three Primary Signatures.

TSi	**Tree structure**
tsi_02_v0	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\}) + [2\_CS(CS1\_PS1/\{1.0.1\}/\{2.0.1\})]]$
tsi_02_v1	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\}) + [2\_CS(PS1/\{1.0.1\}/\{2.0.1\})]]$
tsi_02_v2	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\}) + [2\_CS(CS1\_PS1/\{1.0.2\}/\{2.0.2\})]]$
tsi_02_v3	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\}) + [2\_CS(PS1/\{1.0.2\}/\{2.0.2\})]]$
tsi_02_v4	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})+[2\_CS(CS1\_PS1/\{1.0.2\}/\{2.0.2\}/RTS(1))]]$
tsi_02_v5	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\}) + [2\_CS(CS1\_PS1/\{1.0.2\}/\{2.0.2\}/ RTS(1,\Delta(0,0,1,0)))]]$

Table D.4: TSis with one Primary Signature and one CounterSignature each.

TSi	**Tree structure**
tsi_03_v0	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\}) + [3\_CS(PS1/\{1.0.1\}/\{2.0.1\})] + [4\_CS(PS2/\{1.0.1\}/\{2.0.1\}) + [7\_CS(CS3/\{1.0.3\}/\{2.0.3\})]]] + [2\_PS(PS2/\{1.0.1\}/\{2.0.1\}) + [5\_CS(CS5/\{1.0.5\}/\{2.0.5\})] + [6\_CS(CS6/\{1.0.6\}/\{2.0.6\})]]$
tsi_03_v1	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\}) + [3\_CS(CS3/\{1.0.1\}/\{2.0.1\})] + [4\_CS(PS2/\{1.0.1\}/\{2.0.1\}) + [7\_CS(CS4/\{1.0.3\}/\{2.0.3\})]]] + [2\_PS(PS2/\{1.0.1\}/\{2.0.1\}) + [5\_CS(CS5/\{1.0.5\}/\{2.0.5\})] + [6\_CS(CS6/\{1.0.6\}/\{2.0.6\})]]$
tsi_03_v2	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\}) + [3\_CS(PS3/\{1.0.1\}/\{2.0.1\})] + [4\_CS(CS4/\{1.0.1\}/\{2.0.1\}) + [7\_CS(CS5/\{1.0.3\}/\{2.0.3\})]]] + [2\_PS(PS3/\{1.0.1\}/\{2.0.1\}) + [5\_CS(CS6/\{1.0.5\}/\{2.0.5\})] + [6\_CS(CS7/\{1.0.6\}/\{2.0.6\})]]$
tsi_04_v0	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\}) + [3\_CS(PS2/\{1.0.2\}/\{2.0.2\})] + [4\_CS(CS1/\{1.0.2\}/\{2.0.2\})]] + [2\_PS(PS2/\{1.0.2\}/\{2.0.2\})]$
tsi_05_v0	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\}) + [3\_CS(CS2/\{1.0.2\}/\{2.0.2\})] + [4\_CS(CS2/\{1.0.2\}/\{2.0.2\})]] + [2\_PS(PS1/\{1.0.1\}/\{2.0.1\}) + [5\_CS(CS2/\{1.0.2\}/\{2.0.2\})] + [6\_CS(CS3/\{1.0.3\}/\{2.0.3\})]]$
tsi_05_v1	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\}) + [3\_CS(CS2/\{1.0.2\}/\{2.0.2\})] + [4\_CS(CS3/\{1.0.2\}/\{2.0.2\})]] + [2\_PS(PS1/\{1.0.1\}/\{2.0.1\}) + [5\_CS(CS2/\{1.0.2\}/\{2.0.2\})] + [6\_CS(CS3/\{1.0.3\}/\{2.0.3\})]]$
tsi_05_v2	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})+[3\_CS(CS3/\{1.0.1\}/\{2.0.1\}/RTS(4))]+ [4\_CS(CS4/\{1.0.2\}/\{2.0.2\})]] + [2\_PS(PS2/\{1.0.1\}/\{2.0.1\}) + [5\_CS(CS5/\{1.0.2\}/\{2.0.2\})] + [6\_CS(CS6/\{1.0.1\}/\{2.0.1\})]]$
tsi_06_v0	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\})] + [2\_PS(PS2/\{1.0.2\}/\{2.0.2\}) + [4\_CS(CS1\_PS2/\{1.0.4\}/\{2.0.4\})]+[5\_CS(CS2\_PS2/\{1.0.5\}/\{2.0.5\})]]+ [3\_PS(PS3/\{1.0.3\}/\{2.0.3\})]$
tsi_06_v1	$[1\_PS(PS1/\{1.0.1\}/\{2.0.1\}/RTS(3,\Delta(0,0,1,0)))] + [2\_PS(PS2/\{1.0.2\}/\{2.0.2\}/RTS(1,\Delta(0,0,1,0))) + [4\_CS(CS1\_PS2/\{1.0.4\}/\{2.0.4\}/RTS(2,\Delta(0,0,1,0)))] + [5\_CS(CS2\_PS2/\{1.0.5\}/\{2.0.5\}/RTS(4,\Delta(0,0,1,0))/ RTS(2,\Delta(0,0,2,0)))]] + [3\_PS(PS3/\{1.0.3\}/\{2.0.3\}/$

	$ATS_{20070717000000-20070717091500})]$

Table D.5: TSis with several levels of depth.

TSi	**Tree structure**
$tsi\_OFEPSP+\_main$	$[1\_PS(origin/\{1.0\}/\{0.1\})]$ + $[2\_PS(receiver/\{1.0\}/\{0.2\}/RTS(1,\Delta(0,5,0,0)))$+ $[3\_CS(origin/\{1.0\}/\{0.3\}/RTS(2,\Delta(0,15,0,0)))$ + $[4\_CS(receiver/\{1.0\}/\{0.4\}/RTS(3,\Delta(0,5,0,0)))$+ $[5\_CS(origin/\{1.0\}/\{0.5\}/RTS(4,\Delta(0,15,0,0)))]]]]$
$tsi\_OFEPSP+\_recovery$	$[1\_PS(origin/\{1.0\}/\{0.1\})]$ + $[2\_PS(receiver/\{1.0\}/\{0.2\}/RTS(1,\Delta(0,5,0,0)))$+ $[3\_CS(origin/\{1.0\}/\{0.3\}/RTS(2,\Delta(0,15,0,0)))$ + $[4\_CS(receiver/\{1.0\}/\{0.4\}/RTS(3,\Delta(0,5,0,0)))$+ $[5\_CS(ttp/\{1.0\}/\{0.5\}/RTS(4,\Delta(0,5,0,0)))]]]]$

Table D.6: TSis for OFEPSP+ protocol (see Chapter 9).

D.3 Set of Signatures for the Test Cases

Next, the SSis that have been used as input for the test cases are defined following the textual representation explained in Section D.1.

SSi	**Tree structure**
ssi_00_v0	$[subjectDN1/1.0.1/2.0.1]$
ssi_00_v1	$[subjectDN1/1.0.20/2.0.1]$
ssi_00_v2	$[subjectDN1/1.0.1/2.0.20]$
ssi_01_v0	$[subjectDN1/1.0.1/2.0.1/20070717000000]$
ssi_01_v1	$[subjectDN1/1.0.1/2.0.1/19970617000000]$
ssi_01_v2	$[subjectDN1/1.0.1/2.0.1/20170817000000]$

Table D.7: SSis with one Primary Signature.

SSi	**Tree structure**
ssi_02_v0	$[subjectDN1/1.0.1/2.0.1/20000717000000]$ + $[subjectDN2/1.0.1/2.0.1/20000717000100]$ + $[subjectDN3/1.0.1/2.0.1/20000717000200]$
ssi_02_v1	$[subjectDN1/1.0.1/2.0.1/20000717000000]$ + $[subjectDN1/1.0.1/2.0.1/20000717000100]$ + $[subjectDN3/1.0.1/2.0.1/20000717000200]$
ssi_02_v2	$[subjectDN1/1.0.1/2.0.1/20000717000000]$ + $[subjectDN1/1.0.1/2.0.1/20000717000100]$ + $[subjectDN1/1.0.1/2.0.1/20000717000200]$
ssi_02_v3	$[subjectDN1/1.0.1/2.0.1/20000717000000]$ + $[subjectDN2/1.0.1/2.0.1/20010717000000]$ + $[subjectDN3/1.0.1/2.0.1/20020717000000]$
ssi_02_v4	$[subjectDN1/1.0.2/2.0.2/20000717000000]$ + $[subjectDN2/1.0.3/2.0.3/20000717000100]$ + $[subjectDN3/1.0.1/2.0.1/20000717000200]$

*ssi*_02_*v*5	[*subjectDN*1/1.0.1/2.0.1/20000717000000] + [*subjectDN*1/1.0.2/2.0.2/20000717000100] + [*subjectDN*3/1.0.3/2.0.3/20000717000200]
*ssi*_02_*v*6	[*subjectDN*1/1.0.1/2.0.1/20000717000000] + [*subjectDN*1/1.0.2/2.0.2/20000717000100] + [*subjectDN*1/1.0.3/2.0.3/20000717000200]
*ssi*_02_*v*7	[*subjectDN*1/1.0.1/2.0.1/20000717000110] + [*subjectDN*2/1.0.1/2.0.1/20000717000100] + [*subjectDN*3/1.0.2/2.0.2/20000717000200]
*ssi*_02_*v*8	[*subjectDN*1/1.0.1/2.0.1/20000717000210] + [*subjectDN*2/1.0.1/2.0.1/20000717000100] + [*subjectDN*3/1.0.2/2.0.2/20000717000200]
*ssi*_02_*v*9	[*subjectDN*1/1.0.1/2.0.1/20090718101000] + [*subjectDN*2/1.0.1/2.0.1/20090718110000] + [*subjectDN*3/1.0.3/2.0.3/20090718120000]

Table D.8: SSis with three Primary Signatures.

SSi	**Tree structure**
*ssi*_03_*v*0	[*subjectDN*1/1.0.1/2.0.1 + [*subjectDN*2/1.0.2/2.0.2]]
*ssi*_03_*v*1	[*subjectDN*1/1.0.1/2.0.1 + [*subjectDN*1/1.0.2/2.0.2]]
*ssi*_04_*v*0	[*subjectDN*1/1.0.1/2.0.1 + [*subjectDN*2/1.0.1/2.0.1] + [*subjectDN*3/1.0.1/2.0.1 + [*subjectDN*4/1.0.3/2.0.3]]] + [*subjectDN*2/1.0.1/2.0.1 + [*subjectDN*5/1.0.5/2.0.5] + [*subjectDN*6/1.0.6/2.0.6]]
*ssi*_05_*v*0	[*subjectDN*1/1.0.1/2.0.1 + [*subjectDN*3/1.0.2/2.0.2] + [*subjectDN*4/1.0.2/2.0.2]] + [*subjectDN*2/1.0.2/2.0.2]
*ssi*_06_*v*0	[*subjectDN*1/1.0.1/2.0.1 + [*subjectDN*2/1.0.2/2.0.2] + [*subjectDN*2/1.0.2/2.0.2]] + [*subjectDN*1/1.0.1/2.0.1 + [*subjectDN*2/1.0.2/2.0.2] + [*subjectDN*3/1.0.3/2.0.3]]
*ssi*_06_*v*1	[*subjectDN*1/1.0.1/2.0.1 + [*subjectDN*2/1.0.2/2.0.2] + [*subjectDN*3/1.0.3/2.0.3]] + [*subjectDN*1/1.0.1/2.0.1 + [*subjectDN*2/1.0.2/2.0.2] + [*subjectDN*3/1.0.3/2.0.3]]
*ssi*_06_*v*2	[*subjectDN*1/1.0.1/2.0.1+[*subjectDN*3/1.0.1/2.0.1/20000101100002]+ [*subjectDN*4/1.0.2/2.0.2/20000101100003]] + [*subjectDN*2/1.0.1/2.0.1+[*subjectDN*5/1.0.2/2.0.2/20000101100000]+ [*subjectDN*6/1.0.1/2.0.1/20000101100001]]
*ssi*_06_*v*3	[*subjectDN*1/1.0.1/2.0.1+[*subjectDN*3/1.0.1/2.0.1/20000101100002]+ [*subjectDN*4/1.0.2/2.0.2/20000101100003]] + [*subjectDN*2/1.0.1/2.0.1+[*subjectDN*5/1.0.2/2.0.2/20000101100001]+ [*subjectDN*6/1.0.1/2.0.1/20000101100000]]
*ssi*_07_*v*0	[*subjectDN*1/1.0.1/2.0.1/20070717090000] + [*subjectDN*2/1.0.2/2.0.2/20070717093000 + [*subjectDN*4/1.0.4/2.0.4/20070717100000] + [*subjectDN*5/1.0.5/2.0.5/20070717103000]] + [*subjectDN*3/1.0.3/2.0.3/20070717083000]

Table D.9: SSis with several levels of depth.

SSi	Tree structure
$ssi\_OFEPSP+\_main$	$[CN = Buyer, O = Internet/1.0/0.1/20090717202700] + [CN = Seller, O = Internet/1.0/0.2/20090717203100 + [CN = Buyer, O = Internet/1.0/0.3/20090717204100 + [CN = Seller, O = Internet/1.0/0.4/20090717204300 + [CN = Buyer, O = Internet/1.0/0.5/20090717205000]]]]$
$ssi\_OFEPSP+\_main2$	$[CN = Buyer, O = Internet/1.0/0.1/20090717202700] + [CN = Seller, O = Internet/1.0/0.2/20090717203100 + [CN = Buyer, O = Internet/1.0/0.3/20090717205100 + [CN = Seller, O = Internet/1.0/0.4/20090717205300 + [CN = Buyer, O = Internet/1.0/0.5/20090717206000]]]]$
$ssi\_OFEPSP+\_recovery$	$[CN = Buyer, O = Internet/1.0/0.1/20090717202700] + [CN = Seller, O = Internet/1.0/0.2/20090717203100 + [CN = Buyer, O = Internet/1.0/0.3/20090717204100 + [CN = Seller, O = Internet/1.0/0.4/20090717204300 + [CN = TTP, O = Internet/1.0/0.5/20090717204500]]]]$

Table D.10: SSis for OFEPSP+ protocol (see Chapter 9).

D.4 Test Cases

Next Tables D.11, D.12[1], D.13 and D.14 collect the test cases executed to validate the correct design and implementation of the algorithm. Each test case includes the SSi and TSi evaluated, the test result (*OK*, if the SSi satisfies the TSi, *FAIL* otherwise) and the reason in case the SSi does not satisfy the TSi. It should be mentioned that more test cases than those shown in the Tables have been executed (188 in total), but have not been included in the Tables if the execution path, result and reason did not differ from the included test cases.

SSi	TSi	Result	Reason
ssi_00_v0	tsi_00_v0	OK	
	tsi_00_v1	FAIL	ATS not fulfilled (no *signingTime*)
	tsi_01_v0	FAIL	SSi and TSi have different distributions
	tsi_01_v1	FAIL	SSi and TSi have different distributions
	tsi_01_v2	FAIL	SSi and TSi have different distributions
	tsi_02_v0	FAIL	Deadlock (distribution-based pruning)
	tsi_02_v1	FAIL	Deadlock (distribution-based pruning)
	tsi_02_v2	FAIL	Deadlock (distribution-based pruning)
	tsi_02_v3	FAIL	Deadlock (distribution-based pruning)
	tsi_03_v0	FAIL	SSi and TSi have different distributions
ssi_00_v1	tsi_00_v0	FAIL	Deadlock when exploring signer $subjectDN1$ (due to *sp*)

[1]Though test cases from tsi_01_v3 to tsi_01_v11 take SSis and TSis with the same number of nodes in the single level, the distribution differs due to the existent dimensions for each distribution. Thus, SSi and TSi are not structurally equal. Same reason applies to other test cases in other tables.

	tsi_00_v1	FAIL	Deadlock when exploring signer $subjectDN1$ (due to sp)
	tsi_03_v0	FAIL	SSi and TSi have different distributions
ssi_00_v2	tsi_00_v0	FAIL	Deadlock when exploring signer $subjectDN1$ (due to ct)
	tsi_00_v1	FAIL	Deadlock when exploring signer $subjectDN1$ (due to ct)
ssi_01_v0	tsi_00_v0	OK	
	tsi_00_v1	OK	
ssi_01_v1	tsi_00_v0	OK	
	tsi_00_v1	FAIL	ATS not fulfilled ($signingTime$ before allowed)
ssi_01_v2	tsi_00_v0	OK	
	tsi_00_v1	FAIL	ATS not fulfilled ($signingTime$ after allowed)

Table D.11: Test cases defined for SSis with one level of depth and one Primary Signatures.

SSi	**TSi**	**Result**	**Reason**
ssi_02_v0	tsi_00_v0	FAIL	SSi and TSi have different distributions
	tsi_00_v1	FAIL	SSi and TSi have different distributions
	tsi_01_v0	OK	
	tsi_01_v1	OK	
	tsi_01_v2	OK	
	tsi_01_v3	FAIL	SSi and TSi have different distributions
	tsi_01_v4	FAIL	SSi and TSi have different distributions
	tsi_01_v5	FAIL	SSi and TSi have different distributions
	tsi_01_v6	FAIL	SSi and TSi have different distributions
	tsi_01_v7	FAIL	SSi and TSi have different distributions
	tsi_01_v8	FAIL	SSi and TSi have different distributions
	tsi_01_v9	FAIL	SSi and TSi have different distributions
	tsi_01_v10	FAIL	SSi and TSi have different distributions
	tsi_01_v11	FAIL	SSi and TSi have different distributions
	tsi_01_v12	FAIL	Deadlock when exploring signer $subjectDN2$ (signer $subjectDN1$ was definitely assigned to node 1_PS)
	tsi_01_v13	FAIL	Deadlock when exploring signer $subjectDN2$ (signer $subjectDN1$ was definitely assigned to node 1_PS)
	tsi_01_v14	FAIL	Deadlock when exploring signer $subjectDN2$ (signer $subjectDN1$ was definitely assigned to node 1_PS)
	tsi_02_v0	FAIL	SSi and TSi have different distributions
	tsi_02_v1	FAIL	SSi and TSi have different distributions
	tsi_02_v2	FAIL	SSi and TSi have different distributions
	tsi_02_v3	FAIL	SSi and TSi have different distributions
	tsi_03_v0	FAIL	SSi and TSi have different distributions
ssi_02_v1	tsi_00_v0	FAIL	SSi and TSi have different distributions
	tsi_00_v1	FAIL	SSi and TSi have different distributions
	tsi_01_v0	FAIL	SSi and TSi have different distributions
	tsi_01_v1	FAIL	SSi and TSi have different distributions

	tsi_01_v2	FAIL	SSi and TSi have different distributions
	tsi_01_v3	OK	
	tsi_01_v4	OK	
	tsi_01_v5	OK	
	tsi_01_v6	OK	
	tsi_01_v7	FAIL	RTS not fulfilled
	tsi_01_v8	FAIL	RTS not fulfilled
	tsi_01_v9	FAIL	SSi and TSi have different distributions
	tsi_01_v10	FAIL	SSi and TSi have different distributions
	tsi_01_v11	FAIL	SSi and TSi have different distributions
	tsi_01_v12	FAIL	SSi and TSi have different distributions
	tsi_01_v13	FAIL	SSi and TSi have different distributions
	tsi_01_v14	FAIL	SSi and TSi have different distributions
	tsi_02_v0	FAIL	SSi and TSi have different distributions
	tsi_02_v1	FAIL	SSi and TSi have different distributions
	tsi_02_v2	FAIL	SSi and TSi have different distributions
	tsi_02_v3	FAIL	SSi and TSi have different distributions
	tsi_03_v0	FAIL	SSi and TSi have different distributions
ssi_02_v2	tsi_00_v0	FAIL	SSi and TSi have different distributions
	tsi_00_v1	FAIL	SSi and TSi have different distributions
	tsi_01_v0	FAIL	SSi and TSi have different distributions
	tsi_01_v1	FAIL	SSi and TSi have different distributions
	tsi_01_v2	FAIL	SSi and TSi have different distributions
	tsi_01_v3	FAIL	SSi and TSi have different distributions
	tsi_01_v4	FAIL	SSi and TSi have different distributions
	tsi_01_v5	FAIL	SSi and TSi have different distributions
	tsi_01_v6	FAIL	SSi and TSi have different distributions
	tsi_01_v7	FAIL	SSi and TSi have different distributions
	tsi_01_v8	FAIL	SSi and TSi have different distributions
	tsi_01_v9	OK	
	tsi_01_v10	OK	
	tsi_01_v11	OK	
	tsi_01_v12	FAIL	SSi and TSi have different distributions
	tsi_01_v13	FAIL	SSi and TSi have different distributions
	tsi_01_v14	FAIL	SSi and TSi have different distributions
	tsi_01_v15	FAIL	Deadlock in refinement phase two (node 2_PS)
	tsi_01_v16	FAIL	Deadlock in refinement phase two (node 3_PS)
	tsi_02_v0	FAIL	SSi and TSi have different distributions
	tsi_02_v1	FAIL	SSi and TSi have different distributions
	tsi_02_v2	FAIL	SSi and TSi have different distributions
	tsi_02_v3	FAIL	SSi and TSi have different distributions
	tsi_03_v0	FAIL	SSi and TSi have different distributions
ssi_02_v3	tsi_01_v2	FAIL	RTS not fulfilled
	tsi_01_v14	FAIL	Deadlock when exploring signer $subjectDN2$ (signer $subjectDN1$ was definitely assigned to node 1_PS)
ssi_02_v4	tsi_01_v0	FAIL	Deadlock when exploring signer with sp = 1.0.2 and ct = 2.0.2
	tsi_01_v3	FAIL	SSi and TSi have different distributions
	tsi_01_v4	FAIL	SSi and TSi have different distributions

	*tsi*_01_*v*5	FAIL	SSi and TSi have different distributions
	*tsi*_01_*v*6	FAIL	SSi and TSi have different distributions
	*tsi*_01_*v*7	FAIL	SSi and TSi have different distributions
	*tsi*_01_*v*8	FAIL	SSi and TSi have different distributions
	*tsi*_01_*v*9	FAIL	SSi and TSi have different distributions
	*tsi*_01_*v*10	FAIL	SSi and TSi have different distributions
	*tsi*_01_*v*11	FAIL	SSi and TSi have different distributions
	*tsi*_01_*v*12	OK	
	*tsi*_01_*v*13	FAIL	RTS not fulfilled
	*tsi*_01_*v*14	FAIL	RTS not fulfilled
*ssi*_02_*v*5	*tsi*_01_*v*3	FAIL	Deadlock when exploring signer with *sp* = 1.0.2 and *ct* = 2.0.2
	*tsi*_01_*v*6	FAIL	Deadlock when exploring signer with *sp* = 1.0.2 and *ct* = 2.0.2
*ssi*_02_*v*6	*tsi*_01_*v*9	FAIL	Deadlock when exploring signer with *sp* = 1.0.2 and *ct* = 2.0.2
	*tsi*_01_*v*14	FAIL	SSi and TSi have different distributions
	*tsi*_01_*v*15	OK	
*ssi*_02_*v*7	*tsi*_01_*v*13*b*	OK	
*ssi*_02_*v*8	*tsi*_01_*v*13*b*	FAIL	Deadlock in refinement phase three (node 2_*PS*2)
*ssi*_02_*v*9	*tsi*_01_*v*13*b*	FAIL	Deadlock in refinement phase three

Table D.12: Test cases defined for SSis with one level of depth and three Primary Signatures.

SSi	TSi	Result	Reason
*ssi*_03_*v*0	*tsi*_00_*v*0	FAIL	Deadlock (distribution-based pruning)
	*tsi*_01_*v*9	FAIL	SSi and TSi have different distributions
	*tsi*_02_*v*0	FAIL	Deadlock for countersignature *subjectDN*2
	*tsi*_02_*v*1	FAIL	Deadlock for countersignature *subjectDN*2
	*tsi*_02_*v*2	OK	
	*tsi*_02_*v*3	FAIL	Deadlock for countersignature *subjectDN*2 (signer identifier *PS*1 of child node was definitely assigned to *subjectDN*1)
*ssi*_03_*v*1	*tsi*_02_*v*0	FAIL	Deadlock for countersignature *subjectDN*1
	*tsi*_02_*v*1	FAIL	Deadlock for countersignature *subjectDN*1
	*tsi*_02_*v*2	FAIL	Deadlock for countersignature *subjectDN*1, as its *subjectDN* was definitely assigned to signer identifier *PS*1, and child node has signer identifier *CS*1_*PS*1
	*tsi*_02_*v*3	OK	
*ssi*_04_*v*0	*tsi*_03_*v*0	FAIL	Deadlock when backtracking at first node in first level. Initially assigned nodes' signer identifiers *PS*1 and *PS*2 are definitely assigned to countersignatures *subjectDN*2 and *subjectDN*3. The identifier-based pruning deletes node *PS*1 from the matching. Afterwards, the path-based pruning deletes the remaining matched node *PS*2, and thus there is a deadlock at *subjectDN*1

	tsi_03_v1	FAIL	Deadlock when exploring $S2$ in first level, as $PS2$ is definitely assigned to $subjectDN$. An path-based pruning is performed at signature $S1$ in first level while backtracking. Therefore, a refinement over children nodes is produced before analyzing child $subjectDN3$ of $subjectDN1$
	tsi_03_v2	OK	
	tsi_04_v0	FAIL	Deadlock for countersignature $subjectDN2$
ssi_05_v0	tsi_04_v0	FAIL	Deadlock in refining phase one and after applying identifier-based pruning during forwarding for signer $subjectDN3$ (signer identifier $PS2$ was definitely assigned to $subjectDN2$)
ssi_06_v0	tsi_05_v0	OK	
ssi_06_v1	tsi_05_v1	FAIL	During refinement phase one, two nodes matched with first $subjectDN2$ are initially pruned from active nodes. Subsequently, the identifier-based pruning prunes matched node *CS3(1.0.2/2.0.2)*, as it is definitely assigned to signer *subjectDN3/sp3/2.0.3*. Furthermore, at path-based pruning, and basing on active nodes array, one node remains (6_CS3). That happens symmetrically for the other branch. As a result, two first level signatures are both matched with the same node 2. During refinement phase two, the deadlock is detected (node 1_PS has no visits)
ssi_06_v2	tsi_05_v2	OK	
ssi_06_v3	tsi_05_v2	FAIL	Deadlock while refining in phase One and after pathBasedPruning (forwarding)
ssi_07_v0	tsi_06_v0	OK	
	tsi_06_v1	OK	

Table D.13: Test cases defined for SSis with several levels of depth.

SSi	TSi	Result	Reason
$ssi\_OFEPSP+\_main$	$tsi\_OFEPSP+\_main$	OK	
	$tsi_OFEPSP+\_recovery$	FAIL	Deadlock for signer $CN = Buyer, O = Internet$ when exploring its last signature (NRE)
$ssi\_OFEPSP+\_recovery$	$tsi\_OFEPSP+\_main$	FAIL	Deadlock for signer $CN = TTP, O = Internet$ when exploring its signature (NRE_{TTP})
	$tsi\_OFEPSP+\_recovery$	OK	

$ssi\_OFEPSP+\_main2$	$tsi\_OFEPSP+\_main$	FAIL	RTS not fulfilled for second signature of $CN = Buyer, O = Internet$

Table D.14: Test cases defined for OFEPSP+ protocol (see Chapter 9).

Appendix E

ASN.1 and XML Schemas

E.1 Extended Signature Policy ASN.1 definition

```
ETS-ExtendedElectronicSignaturePolicies-97Syntax { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-mod(0) 30 }

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

-- EXPORTS All

--IMPORTS

-- =======================================================================
-- Internet X.509 Public Key Infrastructure-Certificate and CRL Profile
-- RFC 2459 or RFC 3280 or RFC 5280
-- =======================================================================
--AlgorithmIdentifier, GeneralNames, DirectoryString

--  FROM PKIX1Explicit88
--    { iso(1) identified-organization(3) dod(6) internet(1) security(5)
--      mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit(18) }

--FROM PKIX1Explicit93
--{iso(1) identified-organization(3) dod(6) internet(1) security(5)
--     mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit-93(3)}

-- =======================================================================
-- CAdES: RFC 5126
-- =======================================================================
--SigPolicyQualifierInfo

--   FROM PKIXCAdES08
--    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
--      smime(16) id-aa(2) 15 }

-- ======================================================================
-- Electronic Signature Policies : RFC 3125
-- ======================================================================
--SelectedCommitmentTypes, SigningPeriod, SignPolExtensions, DeltaTime

--FROM ETS-ElectronicSignaturePolicies-97Syntax
--{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
--      smime(16) id-mod(0) 8}
--;

-- ======================================================================
-- S/MIME Object Identifier arcs used in the present document
-- ======================================================================

-- S/MIME OID arc used in the present document
-- id-smime OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
-- rsadsi(113549) pkcs(1) pkcs-9(9) 16 }

-- S/MIME Arcs
-- id-mod OBJECT IDENTIFIER ::= { id-smime 0 }
-- modules
-- id-spq OBJECT IDENTIFIER ::= { id-smime 5 }
-- signature policy qualifier
-- id-cti OBJECT IDENTIFIER ::= { id-smime 6 }
-- commitment type identifier

-- ======================================================================

AlgorithmIdentifier  ::=  SEQUENCE  {
  algorithm          OBJECT IDENTIFIER,
  parameters         ANY DEFINED BY algorithm OPTIONAL  }

GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName

GeneralName ::= CHOICE {
  --otherName                 [0]     OtherName,
  rfc822Name                  [1]     IA5String,
  dNSName                     [2]     IA5String,
  --x400Address               [3]     ORAddress,
  -- directoryName            [4]     Name,
  -- ediPartyName             [5]     EDIPartyName,
  uniformResourceIdentifier   [6]     IA5String,
  iPAddress                   [7]     OCTET STRING,
  registeredID                [8]     OBJECT IDENTIFIER }

DirectoryString ::= CHOICE {
  teletexString          TeletexString (SIZE (1..MAX)),
  printableString        PrintableString (SIZE (1..MAX)),
  universalString        UniversalString (SIZE (1..MAX)),
  utf8String             UTF8String (SIZE (1..MAX)),
  bmpString              BMPString (SIZE (1..MAX)) }

-- ===================================================================

SigPolicyQualifierInfo ::= SEQUENCE {
  sigPolicyQualifierId  SigPolicyQualifierId,
  sigQualifier          ANY DEFINED BY sigPolicyQualifierId }

SigPolicyQualifierId ::= OBJECT IDENTIFIER

id-spq-ets-uri OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840)
rsadsi(113549) pkcs(1) pkcs9(9) smime(16) id-spq(5) 1}

SPuri ::= IA5String

id-spq-ets-unotice OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840)
rsadsi(113549) pkcs(1) pkcs9(9) smime(16) id-spq(5) 2}

SPUserNotice ::= SEQUENCE {
  noticeRef        NoticeReference OPTIONAL,
  explicitText     DisplayText OPTIONAL}

NoticeReference ::= SEQUENCE {
  organization     DisplayText,
  noticeNumbers    SEQUENCE OF INTEGER }
```

```
DisplayText ::= CHOICE {
  visibleString    VisibleString  (SIZE (1..200)),
  bmpString        BMPString      (SIZE (1..200)),
  utf8String       UTF8String     (SIZE (1..200)) }

-- ===========================================================

SelectedCommitmentTypes ::= SEQUENCE OF CHOICE {
  empty NULL,
  recognizedCommitmentType CommitmentType }

CommitmentType ::= SEQUENCE {
  identifier CommitmentTypeIdentifier,
  fieldOfApplication [0] FieldOfApplication OPTIONAL,
  semantics [1] DirectoryString OPTIONAL }

CommitmentTypeIdentifier ::= OBJECT IDENTIFIER

FieldOfApplication ::= DirectoryString

SigningPeriod ::= SEQUENCE {
  notBefore       GeneralizedTime,
  notAfter        GeneralizedTime OPTIONAL }

SignPolExtensions ::= SEQUENCE OF SignPolExtn

SignPolExtn ::= SEQUENCE {
  extnID      OBJECT IDENTIFIER,
  extnValue   OCTET STRING  }

DeltaTime ::= SEQUENCE {
  deltaSeconds    INTEGER,
  deltaMinutes    INTEGER,
  deltaHours      INTEGER,
  deltaDays       INTEGER }

-- ===========================================================
-- Extended Signature Policy Specification
-- ===========================================================

ExtSignaturePolicy ::= SEQUENCE {
  extSignPolicyInfo       ExtSignPolicyInfo,
  extSignPolicyProtection ExtSignPolicyProtection OPTIONAL
}

ExtSignPolicyInfo ::= SEQUENCE {
  extSignPolicyIdentifier ExtSignPolicyIdentifier,
  extSignValidationPolicy ExtSignValidationPolicy,
  extSignContext          [0] ExtSignContext OPTIONAL,
  extSignPolExtensions    [1] SignPolExtensions OPTIONAL
}

ExtSignPolicyProtection ::= SEQUENCE {
  protectionAlg AlgorithmIdentifier,
  protection    BIT STRING
}

ExtSignPolicyIdentifier ::= SEQUENCE {
  extSignPolicyId        ExtSignPolicyId,
  dateOfIssue            GeneralizedTime,
  policyIssuerName       GeneralNames,
  extSigPolicyQualifiers [0] SEQUENCE SIZE (1..MAX) OF SigPolicyQualifierInfo
                         OPTIONAL,
  extSignPolExtensions   [1] SignPolExtensions OPTIONAL
}

ExtSignPolicyId ::= OBJECT IDENTIFIER

ExtSignValidationPolicy ::= SEQUENCE {
  signingPeriod        [0] SigningPeriod,
  treesOfSolutions     [1] TreesOfSolutions,
  extSignPolExtensions [2] SignPolExtensions OPTIONAL
}

TreesOfSolutions ::= SEQUENCE OF treeOfSignature TreeOfSignatures

TreeOfSignatures ::= SEQUENCE OF signature Signature

Signature ::= SEQUENCE {
  identifier             INTEGER (0..MAX),
  signer                 INTEGER (0..MAX),
  acceptableSignPolicies AcceptableSignPolicies,
  allowedCommitmentTypes SelectedCommitmentTypes,
  counterSignatures      [0] TreeOfSignatures OPTIONAL,
  timingAndSequence      [1] TimingAndSequence OPTIONAL,
  extSignPolExtensions   [2] SignPolExtensions OPTIONAL
}

AcceptableSignPolicies ::= SEQUENCE OF signPolicyId SignPolicyId

SignPolicyId ::= OBJECT IDENTIFIER

TimingAndSequence ::= CHOICE {
  absoluteTimingAndSequence [0] SigningPeriod,
  relativeTimingAndSequence [1] SEQUENCE OF RelativeTimingAndSequence
}

RelativeTimingAndSequence ::= SEQUENCE {
  pathToRefSignature SEQUENCE OF INTEGER,
  maxDelta           DeltaTime OPTIONAL
}

ExtSignContext ::= SEQUENCE {
  businessApplicationDomain [0] SigPolicyQualifierInfo OPTIONAL,
  transactionalContext      [1] SigPolicyQualifierInfo OPTIONAL,
  disputeResolution         [2] SigPolicyQualifierInfo OPTIONAL,
  audienceConditions        [3] SigPolicyQualifierInfo OPTIONAL,
  extSignPolExtensions      [4] SignPolExtensions OPTIONAL
}

END
```

E.2 Extended Signature Policy XML definition

Please note that the XSD definition has been obtained using a tool that automatically translates ASN.1 to XML schema. A clearer representation can be obtained, and some elements definitions can be substituted by elements defined in XAdES or other XML related standards (e.g. ObjectIdentifier element type).

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://jlopez.thesis.uc3m.es/ETS-
            ExtendedElectronicSignaturePolicies-97Syntax"
            targetNamespace="http://jlopez.thesis.uc3m.es/ETS-
```

```
          ExtendedElectronicSignaturePolicies-97Syntax"
xmlns:asn1="http://www.obj-sys.com/v1.0/XMLSchema"
elementFormDefault="qualified">
  <xsd:import namespace="http://www.obj-sys.com/v1.0/XMLSchema"
   schemaLocation="http://www.obj-sys.com/v1.0/XMLSchema/asn1.xsd"/>

  <!-- PDU definition -->
  <xsd:element name="sPuri" type="SPuri"/>
  <xsd:simpleType name="SPuri">
     <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <!-- PDU definition -->
  <xsd:element name="sPUserNotice" type="SPUserNotice"/>
  <xsd:complexType name="SPUserNotice">
     <xsd:sequence>
        <xsd:element name="noticeRef" minOccurs="0" type="NoticeReference"/>
        <xsd:element name="explicitText" minOccurs="0" type="DisplayText"/>
     </xsd:sequence>
  </xsd:complexType>

  <!-- PDU definition -->
  <xsd:element name="extSignaturePolicy" type="ExtSignaturePolicy"/>
  <xsd:complexType name="ExtSignaturePolicy">
     <xsd:sequence>
        <xsd:element name="extSignPolicyInfo" type="ExtSignPolicyInfo"/>
        <xsd:element name="extSignPolicyProtection" minOccurs="0"
        type="ExtSignPolicyProtection"/>
     </xsd:sequence>
  </xsd:complexType>

  <xsd:simpleType name="SigPolicyQualifierId">
     <xsd:restriction base="asn1:ObjectIdentifier"/>
  </xsd:simpleType>

  <xsd:simpleType name="CommitmentTypeIdentifier">
     <xsd:restriction base="asn1:ObjectIdentifier"/>
  </xsd:simpleType>

  <xsd:complexType name="DirectoryString">
     <xsd:choice>
```

```
      <xsd:element name="teletexString">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:minLength value="1"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="printableString">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:minLength value="1"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="universalString">
         <xsd:simpleType>
            <xsd:restriction base="asn1:UniversalString">
               <xsd:minLength value="1"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="utf8String">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:minLength value="1"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="bmpString">
         <xsd:simpleType>
            <xsd:restriction base="asn1:BMPString">
               <xsd:minLength value="1"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
   </xsd:choice>
</xsd:complexType>

<xsd:complexType name="FieldOfApplication">
   <xsd:complexContent>
      <xsd:extension base="DirectoryString"/>
   </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="ExtSignPolicyId">
   <xsd:restriction base="asn1:ObjectIdentifier"/>
</xsd:simpleType>

<xsd:simpleType name="SignPolicyId">
   <xsd:restriction base="asn1:ObjectIdentifier"/>
</xsd:simpleType>

<xsd:complexType name="AlgorithmIdentifier">
   <xsd:sequence>
      <xsd:element name="algorithm" type="asn1:ObjectIdentifier"/>
      <xsd:element name="parameters" minOccurs="0" type="asn1:OpenType"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="GeneralName">
   <xsd:choice>
      <xsd:element name="rfc822Name" type="xsd:string"/>
      <xsd:element name="dNSName" type="xsd:string"/>
      <xsd:element name="uniformResourceIdentifier" type="xsd:string"/>
      <xsd:element name="iPAddress" type="xsd:hexBinary"/>
      <xsd:element name="registeredID" type="asn1:ObjectIdentifier"/>
   </xsd:choice>
</xsd:complexType>

<xsd:complexType name="GeneralNames">
   <xsd:sequence minOccurs="1" maxOccurs="unbounded">
      <xsd:element name="GeneralName" type="GeneralName"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SigPolicyQualifierInfo">
   <xsd:sequence>
      <xsd:element name="sigPolicyQualifierId" type="SigPolicyQualifierId"/>
      <xsd:element name="sigQualifier" type="asn1:OpenType"/>
   </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="DisplayText">
   <xsd:choice>
      <xsd:element name="visibleString">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:minLength value="1"/>
               <xsd:maxLength value="200"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="bmpString">
         <xsd:simpleType>
            <xsd:restriction base="asn1:BMPString">
               <xsd:minLength value="1"/>
               <xsd:maxLength value="200"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="utf8String">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:minLength value="1"/>
               <xsd:maxLength value="200"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
   </xsd:choice>
</xsd:complexType>

<xsd:complexType name="NoticeReference">
   <xsd:sequence>
      <xsd:element name="organization" type="DisplayText"/>
      <xsd:element name="noticeNumbers">
         <xsd:simpleType>
            <xsd:list itemType="xsd:integer"/>
         </xsd:simpleType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CommitmentType">
```

```
    <xsd:sequence>
      <xsd:element name="identifier" type="CommitmentTypeIdentifier"/>
      <xsd:element name="fieldOfApplication" minOccurs="0"
      type="FieldOfApplication"/>
      <xsd:element name="semantics" minOccurs="0" type="DirectoryString"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="SelectedCommitmentTypes">
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="CHOICE">
        <xsd:complexType>
          <xsd:choice>
            <xsd:element name="empty" type="asn1:NULL"/>
            <xsd:element name="recognizedCommitmentType"
            type="CommitmentType"/>
          </xsd:choice>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="SigningPeriod">
    <xsd:sequence>
      <xsd:element name="notBefore" type="xsd:dateTime"/>
      <xsd:element name="notAfter" minOccurs="0" type="xsd:dateTime"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="SignPolExtn">
    <xsd:sequence>
      <xsd:element name="extnID" type="asn1:ObjectIdentifier"/>
      <xsd:element name="extnValue" type="xsd:hexBinary"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="SignPolExtensions">
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="SignPolExtn" type="SignPolExtn"/>
    </xsd:sequence>
  </xsd:complexType>
```

```
<xsd:complexType name="DeltaTime">
   <xsd:sequence>
      <xsd:element name="deltaSeconds" type="xsd:integer"/>
      <xsd:element name="deltaMinutes" type="xsd:integer"/>
      <xsd:element name="deltaHours" type="xsd:integer"/>
      <xsd:element name="deltaDays" type="xsd:integer"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ExtSignPolicyIdentifier">
   <xsd:sequence>
      <xsd:element name="extSignPolicyId" type="ExtSignPolicyId"/>
      <xsd:element name="dateOfIssue" type="xsd:dateTime"/>
      <xsd:element name="policyIssuerName" type="GeneralNames"/>
      <xsd:element name="extSigPolicyQualifiers" minOccurs="0">
         <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="unbounded">
               <xsd:element name="SigPolicyQualifierInfo"
               type="SigPolicyQualifierInfo"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="extSignPolExtensions" minOccurs="0"
      type="SignPolExtensions"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="AcceptableSignPolicies">
   <xsd:list itemType="SignPolicyId"/>
</xsd:simpleType>

<xsd:complexType name="RelativeTimingAndSequence">
   <xsd:sequence>
      <xsd:element name="pathToRefSignature">
         <xsd:simpleType>
            <xsd:list itemType="xsd:integer"/>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="maxDelta" minOccurs="0" type="DeltaTime"/>
   </xsd:sequence>
```

```
</xsd:complexType>

<xsd:complexType name="TimingAndSequence">
   <xsd:choice>
      <xsd:element name="absoluteTimingAndSequence" type="SigningPeriod"/>
      <xsd:element name="relativeTimingAndSequence">
         <xsd:complexType>
            <xsd:sequence minOccurs="0" maxOccurs="unbounded">
               <xsd:element name="RelativeTimingAndSequence"
               type="RelativeTimingAndSequence"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
   </xsd:choice>
</xsd:complexType>

<xsd:complexType name="Signature">
   <xsd:sequence>
      <xsd:element name="identifier">
         <xsd:simpleType>
            <xsd:restriction base="xsd:unsignedInt">
               <xsd:minInclusive value="0"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="signer">
         <xsd:simpleType>
            <xsd:restriction base="xsd:unsignedInt">
               <xsd:minInclusive value="0"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="acceptableSignPolicies"
      type="AcceptableSignPolicies"/>
      <xsd:element name="allowedCommitmentTypes"
      type="SelectedCommitmentTypes"/>
      <xsd:element name="counterSignatures" minOccurs="0"
      type="TreeOfSignatures"/>
      <xsd:element name="timingAndSequence" minOccurs="0"
      type="TimingAndSequence"/>
      <xsd:element name="extSignPolExtensions" minOccurs="0"
      type="SignPolExtensions"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="TreeOfSignatures">
   <xsd:sequence minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="signature" type="Signature"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="TreesOfSolutions">
   <xsd:sequence minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="treeOfSignature" type="TreeOfSignatures"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ExtSignValidationPolicy">
   <xsd:sequence>
      <xsd:element name="signingPeriod" type="SigningPeriod"/>
      <xsd:element name="treesOfSolutions" type="TreesOfSolutions"/>
      <xsd:element name="extSignPolExtensions" minOccurs="0"
      type="SignPolExtensions"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ExtSignContext">
   <xsd:sequence>
      <xsd:element name="businessApplicationDomain" minOccurs="0"
      type="SigPolicyQualifierInfo"/>
      <xsd:element name="transactionalContext" minOccurs="0"
      type="SigPolicyQualifierInfo"/>
      <xsd:element name="disputeResolution" minOccurs="0"
      type="SigPolicyQualifierInfo"/>
      <xsd:element name="audienceConditions" minOccurs="0"
      type="SigPolicyQualifierInfo"/>
      <xsd:element name="extSignPolExtensions" minOccurs="0"
      type="SignPolExtensions"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ExtSignPolicyInfo">
```

```
      <xsd:sequence>
         <xsd:element name="extSignPolicyIdentifier"
         type="ExtSignPolicyIdentifier"/>
         <xsd:element name="extSignValidationPolicy"
         type="ExtSignValidationPolicy"/>
         <xsd:element name="extSignContext" minOccurs="0"
         type="ExtSignContext"/>
         <xsd:element name="extSignPolExtensions" minOccurs="0"
         type="SignPolExtensions"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="ExtSignPolicyProtection">
      <xsd:sequence>
         <xsd:element name="protectionAlg" type="AlgorithmIdentifier"/>
         <xsd:element name="protection" type="asn1:BitString"/>
      </xsd:sequence>
   </xsd:complexType>

</xsd:schema>
```

Appendix F

Extended Signature Policy Example for OFEPSP+

This Appendix provides an example of an instantiation of the extended electronic signature policy defined in Chapter 8. The example has been designed for a transactional context where OFEPSP+ protocol, defined in Chapter 9, must be followed. For simplicity purposes, in this example the binding by environment attestation has not been taken into account. The policy instance has been written in ASN.1, complying with the ASN.1 schema given in Appendix E, and defines two Trees of Signatures (TSi) in the Trees of Solutions field as the potential set of signatures that can make the transaction become effective. In particular, the first TSi (*treesOfSolutions[0]*) corresponds to the set of signatures to be generated in the main protocol of OFEPSP+, while the second TSi (*treesOfSolutions[1]*) corresponds to the set of signatures that must be generated in case the recovery subprotocol is to be executed.

```
extSignaturePolicy {
   extSignPolicyInfo {
      extSignPolicyIdentifier {
         extSignPolicyId = { 1 0 }
         dateOfIssue = 19970717103000
         policyIssuerName[0] {
            uniformResourceIdentifier = jlopez.thesis.uc3m.es/Issuer } }
      extSignValidationPolicy {
         signingPeriod {
            notBefore = 19970717103000
            notAfter = 20170717103000 }
         treesOfSolutions[0][0] {
            identifier = 1
            signer = 10
            acceptableSignPolicies[0] = { 1 0 }
```

```
        allowedCommitmentTypes[0] {
           recognizedCommitmentType {
              identifier = { 0 1 }
              fieldOfApplication {
                 printableString = First Partial Non-repudiation of
                 Origin generated under the context of an extended
                 signature policy } } } }
     treesOfSolutions[0][1] {
        identifier = 2
        signer = 20
        acceptableSignPolicies[0] = { 1 0 }
        allowedCommitmentTypes[0] {
           recognizedCommitmentType {
              identifier = { 0 2 }
              fieldOfApplication {
                 printableString = First Partial Non-repudiation of
                 Receipt generated under the context of an extended
                 signature policy } } }
        counterSignatures[0] {
           identifier = 3
           signer = 10
           acceptableSignPolicies[0] = { 1 0 }
           allowedCommitmentTypes[0] {
              recognizedCommitmentType {
                 identifier = { 0 3 }
                 fieldOfApplication {
                    printableString = Second Partial Non-repudiation of
                    Origin generated under the context of an extended
                    signature policy } } }
           counterSignatures[0] {
              identifier = 4
              signer = 20
              acceptableSignPolicies[0] = { 1 0 }
              allowedCommitmentTypes[0] {
                 recognizedCommitmentType {
                    identifier = { 0 4 }
                    fieldOfApplication {
                       printableString = Second Partial Non-repudiation of
                       Receipt generated under the context of an extended
                       signature policy } } }
              counterSignatures[0] {
```

```
         identifier = 5
         signer = 10
         acceptableSignPolicies[0] = { 1 0 }
         allowedCommitmentTypes[0] {
            recognizedCommitmentType {
               identifier = { 0 5 }
               fieldOfApplication {
                  printableString = Non-repudiation of Evidence
                  generated under the context of an extended
                  signature policy } } }
         timingAndSequence {
            relativeTimingAndSequence {
               pathToRefSignature[0] = 2
               pathToRefSignature[1] = 3
               pathToRefSignature[2] = 4
               maxDelta {
                  deltaSeconds = 0
                  deltaMinutes = 15
                  deltaHours = 0
                  deltaDays = 0 } } } }
      timingAndSequence {
         relativeTimingAndSequence {
            pathToRefSignature[0] = 2
            pathToRefSignature[1] = 3
            maxDelta {
               deltaSeconds = 0
               deltaMinutes = 5
               deltaHours = 0
               deltaDays = 0 } } } }
   timingAndSequence {
      relativeTimingAndSequence {
         pathToRefSignature[0] = 2
         maxDelta {
            deltaSeconds = 0
            deltaMinutes = 15
            deltaHours = 0
            deltaDays = 0 } } } }
timingAndSequence {
   relativeTimingAndSequence {
      pathToRefSignature[0] = 1
      maxDelta {
```

```
                    deltaSeconds = 0
                    deltaMinutes = 5
                    deltaHours = 0
                    deltaDays = 0 } } } }
         treesOfSolutions[1][0] {
            identifier = 1
            signer = 10
            acceptableSignPolicies[0] = { 1 0 }
            allowedCommitmentTypes[0] {
               recognizedCommitmentType {
                  identifier = { 0 1 }
                  fieldOfApplication {
                     printableString = First Partial Non-repudiation of
                     Origin generated under the context of an extended
                     signature policy } } } }
         treesOfSolutions[1][1] {
            identifier = 2
            signer = 20
            acceptableSignPolicies[0] = { 1 0 }
            allowedCommitmentTypes[0] {
               recognizedCommitmentType {
                  identifier = { 0 2 }
                  fieldOfApplication {
                     printableString = First Partial Non-repudiation of
                     Receipt generated under the context of an extended
                     signature policy } } }
            counterSignatures[0] {
               identifier = 3
               signer = 10
               acceptableSignPolicies[0] = { 1 0 }
               allowedCommitmentTypes[0] {
                  recognizedCommitmentType {
                     identifier = { 0 3 }
                     fieldOfApplication {
                        printableString = Second Partial Non-repudiation of
                        Origin generated under the context of an extended
                        signature policy } } }
               counterSignatures[0] {
                  identifier = 4
                  signer = 20
                  acceptableSignPolicies[0] = { 1 0 }
                  allowedCommitmentTypes[0] {
                     recognizedCommitmentType {
                        identifier = { 0 4 }
                        fieldOfApplication {
                           printableString = Second Partial Non-repudiation
                           of  Receipt generated under the context of an
                           extended signature policy } } }
                  counterSignatures[0] {
                     identifier = 5
                     signer = 30
                     acceptableSignPolicies[0] = { 1 0 }
                     allowedCommitmentTypes[0] {
                        recognizedCommitmentType {
                           identifier = { 0 5 }
                           fieldOfApplication {
                              printableString = Non-repudiation of Evidence
                              generated by TTP under the context of an
                              extended signature policy } } }
                     timingAndSequence {
                        relativeTimingAndSequence {
                           pathToRefSignature[0] = 2
                           pathToRefSignature[1] = 3
                           pathToRefSignature[2] = 4 } } }
                  timingAndSequence {
                     relativeTimingAndSequence {
                        pathToRefSignature[0] = 2
                        pathToRefSignature[1] = 3
                        maxDelta {
                           deltaSeconds = 0
                           deltaMinutes = 5
                           deltaHours = 0
                           deltaDays = 0 } } } }
               timingAndSequence {
                  relativeTimingAndSequence {
                     pathToRefSignature[0] = 2
                     maxDelta {
                        deltaSeconds = 0
                        deltaMinutes = 15
                        deltaHours = 0
                        deltaDays = 0 } } } }
            timingAndSequence {
               relativeTimingAndSequence {
                  pathToRefSignature[0] = 1
                  maxDelta {
                     deltaSeconds = 0
                     deltaMinutes = 5
                     deltaHours = 0
                     deltaDays = 0 } } } } } } }
         extSignContext {
            businessApplicationDomain {
               sigPolicyQualifierId = { 1 2 840 113549 1 9 16 5 1 }
               sigQualifier {
                  explicitText {
                     visibleString = Sale of goods/international trade
                     transactions } } }
            transactionalContext {
               sigPolicyQualifierId = { 1 2 840 113549 1 9 16 5 1 }
               sigQualifier {
                  explicitText {
                     visibleString = Purchase Order/Acceptance in relation to
                     a book purchase order made through Alice Bookshop Internet
                     Web page between Alice Bookshop and a client of Alice
                     Bookshop } } }
            disputeResolution {
               sigPolicyQualifierId = { 1 2 840 113549 1 9 16 5 1 }
               sigQualifier {
                  explicitText {
                     visibleString = Any disputes arising under this policy
                     shall be referred to a suitably qualified expert, whose
                     decision shall be final and binding upon the parties,
                     provided that this signature policy imposes the constraints
                     under which any signature created under it shall be valid.
                     The dispute resolution procedure shall be carried out in a
                     European court with appropriate responsibilities } } }
            audienceConditions {
               sigPolicyQualifierId = { 1 2 840 113549 1 9 16 5 1 }
               sigQualifier {
                  explicitText {
                     visibleString = The digital signature-based evidence is
                     only valid in a specified jurisdiction, where laws exist
                     which recognize the legal validity of signatures created
                     under conditions as specified in the policy } } } }
```

Appendix G

High-Level Protocol Specification Language for OFEPSP+

Next, the High-Level Protocol Specification Language (HLPSL) that models the improved Optimistic Fair Exchange Protocol based on Signature Policies (OFEPSP+) is given. It should be mentioned that, for simplicity purposes, the HLPSL specification assumes the binding by procedure, and not the environment attestation technique applied in Chapter 9.

```
% -----------------
% Rol origin (E1)
% -----------------
role e1(
  E1,E2,R,S                 : agent,
  Pub_e1,Pub_e2,Pub_r,Pub_s : public_key,
  SND_E1_R,RCV_E1_R,
  SND_E1_S,RCV_E1_S         : channel(dy),
  Msg                       : text,
  Tpl_id                    : text) played_by E1

def=
  local
    State                   : nat,
    % Unique identifier of the protocol run
    Label                   : text,
    % Evidences generated
    PNRO1,NRE,Abort         : {message}_inv(public_key),
    % Evidences verified
    PNRR2,NRE_TTP,Abort_TTP : {message}_inv(public_key)

  init
```

```
    State := 0

  transition

% Main protocol
0.  State   = 0  /\ RCV_E1_R(start) =|>
    State' := 1  /\ Label' := new()
                 /\ PNRO1' := {Pub_e1.Pub_e2.Pub_r.Pub_s.Msg.Label'.
                    Tpl_id}_inv(Pub_e1)
                 /\ SND_E1_R(PNRO1')
                 /\ witness(E1,R,r_e1_pnro1,PNRO1')

1.  State   = 1  /\ RCV_E1_R(PNRR2')
                 /\ PNRR2' = {{Pub_e1.Pub_e2.Pub_r.Pub_s.{Pub_e1.Pub_e2.
                    Pub_r.Pub_s.Msg.Label.Tpl_id}_inv(Pub_r)}
                    _inv(Pub_e2)}_inv(Pub_r) =|>
    State' := 2  /\ NRE' := {PNRR2'}_inv(Pub_e1)
                 /\ SND_E1_R(NRE')
                 /\ witness(E1,R,r_e1_nre,NRE')
                 /\ wrequest(E1,R,e1_r_pnrr2,PNRR2')

% Abort subprotocol
2.  State   = 1  --|>
    State' := 3  /\ Abort' :=  {abort.Label}_inv(Pub_e1)
                 /\ SND_E1_S(Abort')

% - Protocol already recovered by receiver
3.  State   = 3  /\ RCV_E1_S(NRE_TTP')
                 /\ NRE_TTP' = {{{Pub_e1.Pub_e2.Pub_r.Pub_s.{Pub_e1.Pub_e2.
                    Pub_r.Pub_s.Msg.Label.Tpl_id}_inv(Pub_r)}_inv(Pub_e2)}
                    _inv(Pub_r)}_inv(Pub_s) =|>
    State' := 4

% - Protocol successfully aborted
4.  State   = 3  /\ RCV_E1_S(Abort_TTP')
                 /\ Abort_TTP' = {{abort.Label}_inv(Pub_e1)}_inv(Pub_s) =|>
    State' := 5

end role
```

```
% -----------------
% Rol origin (E2)
% -----------------
role e2(
  E1,E2,R                   : agent,
  Pub_e1,Pub_e2,Pub_r,Pub_s : public_key,
  SND_E2_R,RCV_E2_R         : channel(dy),
  Msg                       : text,
  Tpl_id                    : text) played_by E2

def=
  local
    State       : nat,
    % Unique identifier of the protocol run received from R
    Label       : text,
    % Evidence generated
    PNRO2       : {message}_inv(public_key),
    % Evidences verified
    PNRO1,PNRR1 : {message}_inv(public_key)

  init
    State := 0

  transition

% Main protocol
0.  State   = 0  /\ RCV_E2_R(PNRO1'.PNRR1')
                 /\ PNRO1' = {Pub_e1.Pub_e2.Pub_r.Pub_s.Msg.Label'.Tpl_id}
                    _inv(Pub_e1)
                 /\ PNRR1' = {Pub_e1.Pub_e2.Pub_r.Pub_s.Msg.Label'.Tpl_id}
                    _inv(Pub_r) =|>
    State' := 1  /\ PNRO2' := {Pub_e1.Pub_e2.Pub_r.Pub_s.PNRR1'}
                    _inv(Pub_e2)
                 /\ SND_E2_R(PNRO2')
                 /\ witness(E2,R,r_e2_pnro2,PNRO2')
                 /\ wrequest(E2,R,e2_r_pnrr1,PNRR1')

end role
```

```
% -----------------
% Rol receiver (R)
% -----------------
role receiver(
  E1,E2,R,S                 : agent,
  Pub_e1,Pub_e2,Pub_r,Pub_s : public_key,
  SND_R_E1,RCV_R_E1,
  SND_R_E2,RCV_R_E2,
  SND_R_S,RCV_R_S           : channel(dy),
  Tpl_id                    : text) played_by R

def=
  local
    State                             : nat,
    Label                             : text,
    Msg                               : text,
    % Evidences generated
    PNRR1,PNRR2                       : {message}_inv(public_key),
    % Evidences verified
    PNRO1,PNRO2,NRE,NRE_TTP,Abort_TTP : {message}_inv(public_key)

  init
    State := 0

  transition

% Main protocol
0.  State   = 0  /\ RCV_R_E1(PNRO1')
                 /\ PNRO1' = {Pub_e1.Pub_e2.Pub_r.Pub_s.Msg'.Label'.Tpl_id}
                    _inv(Pub_e1) =|>
    State' := 1  /\ PNRR1' := {Pub_e1.Pub_e2.Pub_r.Pub_s.Msg'.Label'.Tpl_id}
                    _inv(Pub_r)
                 /\ SND_R_E2(PNRO1'.PNRR1')
                 /\ witness(R,E2,e2_r_pnrr1,PNRR1')
                 /\ wrequest(R,E1,r_e1_pnro1,PNRO1')


1.  State   = 1  /\ RCV_R_E2(PNRO2')
                 /\ PNRO2' = {Pub_e1.Pub_e2.Pub_r.Pub_s.PNRR1}_inv(Pub_e2) =|>
    State' := 2  /\ PNRR2' := {PNRO2'}_inv(Pub_r)
                 /\ SND_R_E1(PNRR2')
```

```
                /\ witness(R,E1,e1_r_pnrr2,PNRR2')
                /\ wrequest(R,E2,r_e2_pnro2,PNRO2')

2.  State   = 2  /\ RCV_R_E1(NRE')
                 /\ NRE' = {PNRR2}_inv(Pub_e1) =|>
    State' := 3  /\ wrequest(R,E1,r_e1_nre,NRE')

% Quit
3.  State   = 1  --|>
    State' := 4

% Recovery subprotocol
4.  State   = 2  --|>
    State' := 5  /\ SND_R_S(PNRO1.PNRR1.PNRO2.PNRR2)

% - Protocol successfully recovered
5.  State   = 5  /\ RCV_R_S(NRE_TTP')
                 /\ NRE_TTP' = {PNRR2}_inv(Pub_s) =|>
    State' := 6

% - Protocol already aborted by origin
6.  State   = 5  /\ RCV_R_S(Abort_TTP')
                 /\ Abort_TTP' = {{abort.Label}_inv(Pub_e1)}_inv(Pub_s) =|>
    State' := 7

end role

% -----------------
% Rol server (TTP)
% -----------------
role server(
  S                         : agent,
  Pub_s,Pub_e1,Pub_e2,Pub_r : public_key,
  AList                     : (message) set,
  RList                     : (message) set) played_by S

def=
  local
    State                 : nat,
    Label                 : text,
    Msg                   : text,
    Tpl_id                : text,
    % Evidences generated
    NRE_TTP,Abort_TTP     :  {message}_inv(public_key),
    % Evidences verified
    PNRO1,PNRR1,PNRO2,
    PNRR2,Abort           : {message}_inv(public_key),
    SND,RCV               : channel(dy)

  init
    State := 0

  transition

% State values:
% 0: initial value
% 1: value after an abort
% 2: value after a resolve
% The server can be used for only one session

% Abort subprotocol (E1), when not previously aborted nor recovered
1.  State   = 0  /\ RCV(Abort')
                 /\ Abort' = {abort.Label'}_inv(Pub_e1)
                 /\ not(in({{{Pub_e1.Pub_e2.Pub_r.Pub_s.{Pub_e1.Pub_e2.Pub_r.
                    Pub_s.Msg'.Label'.Tpl_id'}_inv(Pub_r)}_inv(Pub_e2)}
                    _inv(Pub_r)}_inv(Pub_s), RList)) =|>
    State' := 1  /\ Abort_TTP' := {Abort'}_inv(Pub_s)
                 /\ AList' := cons(Abort_TTP', AList)
                 /\ SND(Abort_TTP')

% Recovery subprotocol (R), when not previously aborted nor recovered
2.  State   = 0  /\ RCV(PNRO1'.PNRR1'.PNRO2'.PNRR2')
                 /\ PNRO1' = {Pub_e1.Pub_e2.Pub_r.Pub_s.Msg'.Label'.Tpl_id'}
                    _inv(Pub_e1)
                 /\ PNRR1' = {Pub_e1.Pub_e2.Pub_r.Pub_s.Msg'.Label'.Tpl_id'}
                    _inv(Pub_r)
                 /\ PNRO2' = {Pub_e1.Pub_e2.Pub_r.Pub_s.PNRR1'}_inv(Pub_e2)
                 /\ PNRR2' = {PNRO2'}_inv(Pub_r)
                 /\ not(in({{abort.Label'}_inv(Pub_e1)}_inv(Pub_s), AList)) =|>
    State' := 2  /\ NRE_TTP' := {PNRR2'}_inv(Pub_s)
                 /\ RList' := cons(NRE_TTP', RList)
                 /\ SND(NRE_TTP')
```

```
% Abort subprotocol (E1), when already aborted
3.  State   = 1  /\ RCV(Abort')
                 /\ Abort' = {abort.Label'}_inv(Pub_e1)
                 /\ in({Abort'}_inv(Pub_s), AList) =|>
    State' := 1  /\ Abort_TTP' := {Abort'}_inv(Pub_s)
                 /\ SND(Abort_TTP')

% Recovery subprotocol (R), when previously aborted
4.  State   = 1  /\ RCV(PNRO1'.PNRR1'.PNRO2'.PNRR2')
                 /\ PNRO1' = {Pub_e1.Pub_e2.Pub_r.Pub_s.Msg'.Label'.Tpl_id'}
                    _inv(Pub_e1)
                 /\ PNRR1' = {Pub_e1.Pub_e2.Pub_r.Pub_s.Msg'.Label'.Tpl_id'}
                    _inv(Pub_r)
                 /\ PNRO2' = {Pub_e1.Pub_e2.Pub_r.Pub_s.PNRR1'}_inv(Pub_e2)
                 /\ PNRR2' = {PNRO2'}_inv(Pub_r)
                 /\ in({{abort.Label'}_inv(Pub_e1)}_inv(Pub_s), AList) =|>
    State' := 1  /\ Abort_TTP' := {{abort.Label'}_inv(Pub_e1)}_inv(Pub_s)
                 /\ SND(Abort_TTP')

% Abort subprotocol (E1), when previously recovered
5.  State   = 2  /\ RCV(Abort')
                 /\ Abort' = {abort.Label'}_inv(Pub_e1)
                 /\ in({{{Pub_e1.Pub_e2.Pub_r.Pub_s.{Pub_e1.Pub_e2.Pub_r.
                    Pub_s.Msg.Label'.Tpl_id}_inv(Pub_r)}_inv(Pub_e2)}_
                    inv(Pub_r)}_inv(Pub_s), RList) =|>
    State' := 2  /\ NRE_TTP' := {{{Pub_e1.Pub_e2.Pub_r.Pub_s.
                    {Pub_e1.Pub_e2.Pub_r.Pub_s.Msg.Label'.Tpl_id}_inv(Pub_r)}
                    _inv(Pub_e2)}_inv(Pub_r)}_inv(Pub_s)
                 /\ SND(NRE_TTP')

% Recovery subprotocol (R), when already recovered
6.  State   = 2  /\ RCV(PNRO1'.PNRR1'.PNRO2'.PNRR2')
                 /\ PNRO1' = {Pub_e1.Pub_e2.Pub_r.Pub_s.Msg.Label.Tpl_id}
                    _inv(Pub_e1)
                 /\ PNRR1' = {Pub_e1.Pub_e2.Pub_r.Pub_s.Msg.Label.Tpl_id}
                    _inv(Pub_r)
                 /\ PNRO2' = {Pub_e1.Pub_e2.Pub_r.Pub_s.PNRR1'}_inv(Pub_e2)
                 /\ PNRR2' = {PNRO2'}_inv(Pub_r)
                 /\ in({PNRR2'}_inv(Pub_s), RList) =|>
    State' := 2  /\ NRE_TTP' := {PNRR2'}_inv(Pub_s)
```

```
                /\ SND(NRE_TTP')

end role

% -----------------
% Rol session
% -----------------
role session(
  E1,E2,R,S                 : agent,
  Pub_e1,Pub_e2,Pub_r,Pub_s : public_key,
  Msg                       : text,
  Tpl_id                    : text)

def=
  local
    SND_E1_R,RCV_E1_R,SND_E1_S,RCV_E1_S,
    SND_E2_R,RCV_E2_R,
    SND_R_E1,RCV_R_E1,SND_R_E2,RCV_R_E2,SND_R_S,RCV_R_S : channel(dy)

composition
      e1(E1,E2,R,S,Pub_e1,Pub_e2,Pub_r,Pub_s,SND_E1_R,RCV_E1_R,
      SND_E1_S,RCV_E1_S,Msg,Tpl_id)
  /\  e2(E1,E2,R,Pub_e1,Pub_e2,Pub_r,Pub_s,SND_E2_R,RCV_E2_R,
      Msg,Tpl_id)
  /\  receiver(E1,E2,R,S,Pub_e1,Pub_e2,Pub_r,Pub_s,SND_R_E1,
      RCV_R_E1,SND_R_E2,RCV_R_E2,SND_R_S,RCV_R_S,Tpl_id)

end role

% -----------------
% Rol environment
% -----------------
role environment()

def=

  local
    AList,RList   : (message) set

  const
    e1,e2,r,s                                : agent,
    pub_e1,pub_e2,pub_r,pub_s,pub_i           : public_key,
    % Reference of the template used in the transaction
    % [Pre-shared knowledge between E1, E2 and Receiver]
    tpl_id,tpl_id2,tpl_id3,tpl_id4            : text,
    % Message sent by E1 to R
    % [Pre-shared knowledge between E1 and E2]
    msg,msg2,msg3,msg4                        : text,
    timeout,abort                             : protocol_id,
    e1_r_pnrr2,e1_s_nre_ttp,e1_s_abort_ttp    : protocol_id,
    e2_r_pnrr1,e2_s_nre_ttp,e2_s_abort_ttp    : protocol_id,
    r_e1_pnro1,r_e1_nre,r_e2_pnro2,
    r_s_nre_ttp,r_s_abort_ttp                 : protocol_id,
    s_e1_abort,s_e2_abort,s_r_pnrr1,s_r_pnrr2 : protocol_id

  init
    AList := {} /\
    RList := {}

% Not every message (msg) nor template identifier (tpl_id) are initially
% known by the intruder. Messages 'msg' and 'msg4' are only known by
% origin's E1 and E2, and template identifier 'tpl_id' by
% origin's E1 and E2 and the Receiver
intruder_knowledge = {e1,e2,r,s,pub_e1,pub_e2,pub_r,pub_s,pub_i,
                      inv(pub_i),msg2,msg3,tpl_id2,tpl_id3,tpl_id4}

% It should be tested several instances of origin and receiver and just
% one instance of server. This is done by instanciating the server here,
% and origin and receiver in session role

% Different 'msg' and 'tpl_id' are given in each session
composition
   % Protocol session with legitimate agents
   session(e1,e2,r,s,pub_e1,pub_e2,pub_r,pub_s,msg,tpl_id)

   % Protocol session with legitimate agents playing a different role
   % than expected
   %/\ session(r,e2,e1,s,pub_r,pub_e2,pub_e1,pub_s,msg,tpl_id)

   % Protocol session with legitimate agents playing a different role
   % than expected
   %/\ session(e1,r,e2,s,pub_e1,pub_r,pub_e2,pub_s,msg,tpl_id)
   % Protocol session with legitimate agents playing a different role
   % than expected
   %/\ session(e2,e1,r,s,pub_e2,pub_e1,pub_r,pub_s,msg,tpl_id)

   % 2: protocol session with intruder impersonating e1
   %/\ session(i,e2,r,s,pub_i,pub_e2,pub_r,pub_s,msg2,tpl_id2)

   % 3: protocol session with intruder impersonating e2
   %/\ session(e1,i,r,s,pub_e1,pub_i,pub_r,pub_s,msg3,tpl_id3)

   % 4: protocol session with intruder impersonating r
   %/\ session(e1,e2,i,s,pub_e1,pub_e2,pub_i,pub_s,msg4,tpl_id4)

   /\ server(s,pub_s,pub_e1,pub_e2,pub_r,AList,RList)
   %/\ server(s,pub_s,pub_i,pub_e2,pub_r,AList,RList)

end role

% -----------------
% Security goals
% Internally, temporal logic formulae is used
% -----------------
goal

weak_authentication_on r_e1_pnro1
weak_authentication_on e2_r_pnrr1
weak_authentication_on r_e2_pnro2
weak_authentication_on e1_r_pnrr2
weak_authentication_on r_e1_nre

end goal

% -----------------
% Execution ...
% -----------------
environment()
```