Development of a tool for validating ETSI AdES digital signatures as defined by the European Standard ETSI EN 319 102-1

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

CARLOS CONTRERAS MARTÍNEZ

In partial fulfilment

of the requirements for the degree in

TELECOMMUNICATION TECHNOLOGIES AND SERVICES ENGINEERING

Advisor: Juan Carlos Cruellas Ibarz

Barcelona, October 2021

Abstract

The objectives of the various European standards for digital signatures are to establish common specifications within the European Union on how the creation and validation of these should be carried out. This makes it possible to use interoperable electronic signatures across borders of Europe.

This thesis consists of the development of a tool to validate ETSI AdES digital signatures according to the European standard ETSI EN 319 102-1. For this purpose, a study of the different standards has been carried out, together with Object-Oriented Analysis and Design techniques, to achieve the implementation of the validation algorithm and the development of a unit testing framework to check its correct operation. The result is a tool capable of validating Basic Signatures, Signatures with Time and Signatures with Long-Term Validation Material of any ETSI AdES signature form (XAdES, CAdES and PAdES).

Resum

Els objectius de les diferents normes europees per a les signatures digitals són establir especificacions comuns dins de la Unió Europea sobre com s'ha de dur a terme la creació i validació de les mateixes. Això fa possible l'ús de signatures electròniques interoperables a través de les fronteres d'Europa.

Aquesta tesi consisteix en el desenvolupament d'una eina de validació de signatures digitals ETSI AdES segons la norma europea ETSI EN 319 102-1. Per a això, s'ha realitzat un estudi dels diferents estàndards, juntament amb tècniques d'Anàlisi i Disseny Orientat a Objectes, per aconseguir la implementació de l'algoritme de validació i el desenvolupament d'un marc de proves unitàries per a comprovar el seu correcte funcionament. El resultat és una eina capaç de validar Firmes Bàsiques, Firmes amb Temps i Firmes amb Material de Validació a Llarg Termini de qualsevol format de signatura ETSI AdES (XAdES, CAdES i PAdES).

Resumen

Los objetivos de las distintas normas europeas para las firmas digitales son establecer especificaciones comunes dentro de la Unión Europea sobre cómo debe llevarse a cabo la creación y validación de las mismas. Esto hace posible el uso de firmas electrónicas interoperables a través de las fronteras de Europa.

Esta tesis consiste en el desarrollo de una herramienta de validación de firmas digitales ETSI AdES según la norma europea ETSI EN 319 102-1. Para ello, se ha realizado un estudio de los diferentes estándares, junto con técnicas de Análisis y Diseño Orientado a Objetos, para así lograr la implementación del algoritmo de validación y el desarrollo de un marco de pruebas unitarias para comprobar su correcto funcionamiento. El resultado es una herramienta capaz de validar Firmas Básicas, Firmas con Tiempo y Firmas con Material de Validación a Largo Plazo de cualquier formato de firma ETSI AdES (XAdES, CAdES y PAdES).

Revision history and approval record

Revision	Date	Purpose
0	01/09/2021	Document creation
1	11/09/2021	Document revision
2	21/09/2021	Document revision
3	01/10/2021	Document revision
4	11/10/2021	Document approval

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Carlos Contreras Martínez	carlos.contreras.martinez@estudiantat.upc.edu
Juan Carlos Cruellas Ibarz	cruellas@ac.upc.edu

Written by:		Reviewed and approved by:	
Date	01/09/2021	Date	11/10/2021
Name	Carlos Contreras Martínez	Name	Juan Carlos Cruellas Ibarz
Position	Project Author	Position	Project Supervisor

Table of contents

Abstract 1
Resum 2
Resumen 3
Revision history and approval record 4
Table of contents 5
List of Figures 7
List of Tables: 8
1. Introduction 9
1.1. Objectives 9
1.2. Requirements and specifications 9
1.3. Methods and procedures 10
1.4. Work plan 10
1.4.1. Work Breakdown Structure 10
1.4.2. Work Packages 11
1.4.3. Gantt Diagram 12
1.5. Deviations and incidences 12
2. State of the art of the technology used or applied in this thesis: 13
2.1. What is an electronic signature? 13
2.2. Electronic signature levels 13
2.3. AdES profiles 14
2.4. Electronic certificate 14
2.5. Validation of an AdES signature 15
2.6. Time stamp 16
3. Methodology / project development: 17
3.1. Analysis and Object-Oriented Design 17
3.2. Signature validation 17
3.2.1. Signature validation model 17
3.2.2. Basic Building Blocks 20
3.2.2.1. Identification of the signing certificate 20
3.2.2.2. Validation context initialization 21
3.2.2.3. Revocation freshness checker 22
3.2.2.4. X.509 certificate validation 23
3.2.2.5. PKIX Certification Path Validation 25

3.2.2.6. Cryptographic verification....26
3.2.2.7. Signature acceptance validation....27
3.2.3. Validation process for Basic Signatures....27
3.2.4. Time-stamp validation building block....29
3.2.5. Validation process for Signatures with Time and Signatures with Long-Term Validation Material....29
3.3. Automatic testing framework....31
4. Results....36
4.1. XAdES basic signature validation example....36
5. Budget....38
6. Conclusions and future development:....39
Bibliography:....40
Appendices:....41
Glossary....46

List of Figures

Figure 1: Work Breakdown Structure 10
Figure 2: Gantt Diagram 12
Figure 3: Conceptual Model of Signature Validation 18
Figure 4: Digital Signature 18
Figure 5: Signature Lifecycle 19
Figure 6: Basic Signature Validation 20
Figure 7: Identification of the signing certificate 21
Figure 8: Validation context initialization 22
Figure 9: Revocation freshness checker 23
Figure 10: X.509 certificate validation 24
Figure 11: PKIX Certification Path Validation 25
Figure 12: Cryptographic verification 26
Figure 13: Signature acceptance validation 27
Figure 14: Validation process for Basic Signatures 28
Figure 15: Time-stamp validation building block 29
Figure 16: Validation process for Signatures with Time and Signatures with Long-Term Validation Material 30
Figure 17: UML diagram of an AdES signature 41
Figure 18: UML diagram of the validation system 42
Figure 19: Signer's Document example 43
Figure 20: Basic XAdES signature example 45
Figure 21: Dummy certificate 45

List of Tables:

Table 1: Set of unit tests 35
Table 2: Budget 38

1. Introduction

The continuous growth of electronic commerce leads us to need suitable security controls and mechanisms to protect its transactions and guarantee the trust of its users, since this is essential for its success and constant development. In this sense, electronic signatures are an important security component that can be used in a variety of situations.

In general, they are oriented to carry out operations over the Internet that in everyday life require a signature to validate them. They provide three characteristics in Internet communication: identification of the signer, data integrity and non-repudiation. But apart from that, the practical applications of it are many and varied:

- Contracts (sales, employment, lease, insurance, etc.)
- Transactions (e-commerce, online banking, etc.)
- Administrative procedures (tax declarations, requests for birth certificates, etc.)

1.1. Objectives

The European Standard ETSI EN 319 102-1 standardized a procedure for validating ETSI AdES digital signatures. AdES signatures are defined in a way that allow to check their validity long time after having been generated, even when some of the supporting certificates have expired or have been revoked, or also after the keys and cryptographic algorithms used for their generation have broken.

The purpose of this project is to design and implement a software tool prototype capable of validating some forms of ETSI AdES signatures together with an automatic testing framework for the prototype.

The project main goals are:

- Acquire good knowledge of ETSI AdES signatures and their validation procedure, defined within EN 319 102-1.
- Acquire good knowledge of at least one AdES format. Mainly, the XAdES format, defined in the ETSI EN 319 132-1 standard.
- Acquire good knowledge of some basic techniques of Object-Oriented Analysis and Design and some few software design patterns.
- Design an Object-Oriented software system meeting the requirements defined by ETSI EN 319 102-1.
- Implement a Java prototype tool capable of validating some forms of ETSI AdES signatures, especially XAdES signatures.
- Acquire good knowledge of testing frameworks, particularly JUnit and Mockito.
- Design and implement an automatic testing framework for the aforementioned prototype.

1.2. Requirements and specifications

Project requirements:

- The validation tool prototype must be able to validate signatures in the XAdES form.
- The software system must comply with the requirements defined in the ETSI EN 319 102-1 standard.
- The software must include an automated testing infrastructure that performs Unit tests. This element shall be critical for a good maintenance.

Project specifications:

- Validation tool in Java language.
- Object-Oriented software system.
- Ability to validate Basic Signatures, Signatures with Time and Signatures with Long-Term Validation Material, according to the ETSI EN 319 102-1 standard.
- Use of the following Java libraries: BouncyCastle, JUnit and Mockito.
- Standards and technical specifications used for the study:
    - ETSI EN 319 132 XML Advanced Electronic Signatures (XAdES).
    - ETSI EN 319 122 CMS Advanced Electronic Signatures (CAdES).
    - ETSI EN 319 142 PDF Advanced Electronic Signature Profiles (PAdES).
    - ETSI EN 319 162 Associated Signature Containers (ASiC).
    - ETSI TS 119 102-1 Procedures for Creation and Validation of AdES Digital Signatures.

1.3. Methods and procedures

The project was born as the idea of the professor and supervisor Juan Carlos Cruellas, who is a member of the Committee of Electronic Signatures and Infrastructure (ESI) of ETSI (European Telecommunications Standards Institute) to whom the European Commission recognizes the ability to generate European Technical Standards in the field of the electronic signature. He is also the author of AdES signature conformance checking tools.

As a member of the committee that published the ETSI EN 319 102-1 standard, he has also established the first basics of the design of a validation tool. But in general terms, the project starts from scratch, since currently the UPC research group does not have any tool that implements the aforementioned standard.

1.4. Work plan

1.4.1. Work Breakdown Structure

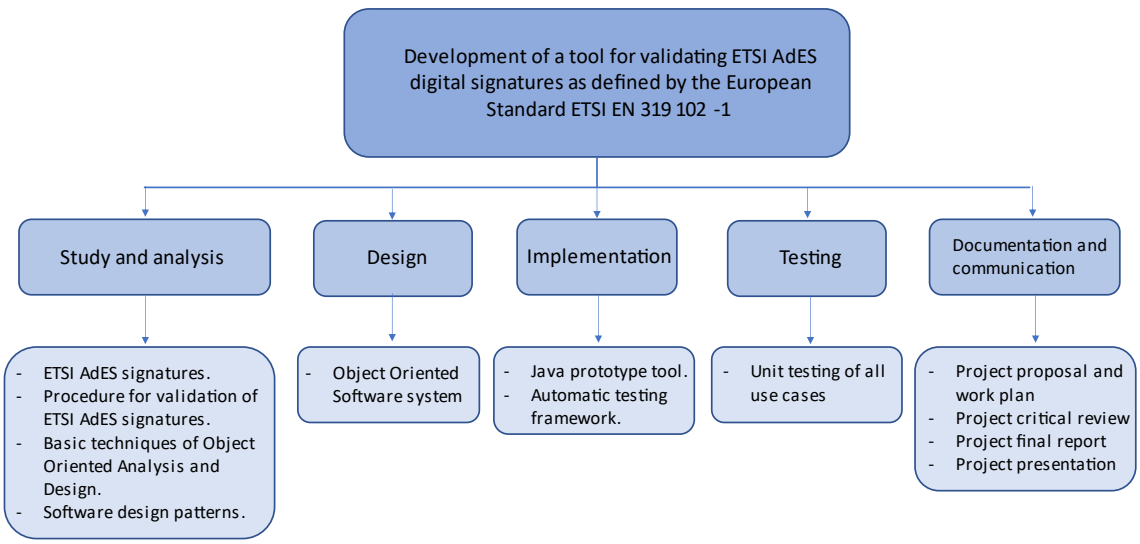

Figure 1: Work Breakdown Structure

1.4.2. Work Packages

Project: Study and Analysis	WP ref: WP1
Major constituent: Acquire information	Sheet 1 of 5
Short description: Gather information, understand how electronic signatures and their standards work, learn software design techniques	Start date: 23/02/2021 End date: 23/03/2021
Internal task T1: Study the basics of ETSI AdES signatures. Internal task T2: Study the procedure for validation of ETSI AdES signatures defined within EN 319 102-1. Internal task T3: Study basic techniques of Object-Oriented Analysis and Design. Internal task T4: Study some few software design patterns.	Deliverables: Analysis of the documents studied.

Project: Design	WP ref: WP2
Major constituent: Software design	Sheet 2 of 5
Short description: Design the structure of the validation tool based on the techniques learned.	Start date: 23/03/2021 End date: 11/05/2021
Internal task T1: Design a OO software system meeting the requirements defined by ETSI EN 319 102-1.	Deliverables: Structure of the software to create.

Project: Implementation	WP ref: WP3
Major constituent: Software development	Sheet 3 of 5
Short description: Create the necessary code from the planned design.	Start date: 06/04/2021 End date: 27/07/2021
Internal task T1: Implement a Java prototype tool able to validate some forms of ETSI AdES signatures.	Deliverables: Software tool.

Project: Testing	WP ref: WP4
Major constituent: Simulation	Sheet 4 of 5
Short description: Create a testing tool.	Start date: 25/05/2021 End date: 03/08/2021
Internal task T1: Design and implement an automatic testing framework for the aforementioned prototype.	Deliverables: Results of the validation tool.

Project: Documentation and Communication	WP ref: WP5
Major constituent: Document process	Sheet 5 of 5
Short description: Develop procedures and reviews to keep track of the project.	Start date: 26/02/2021 End date: 11/10/2021
Internal task T1: Document the analysis, software system design and testing framework. Internal task T2: Project Proposal and Work Plan. Internal task T3: Critical Review. Internal task T4: Final Review.	Deliverables: Documentation.

1.4.3. Gantt Diagram

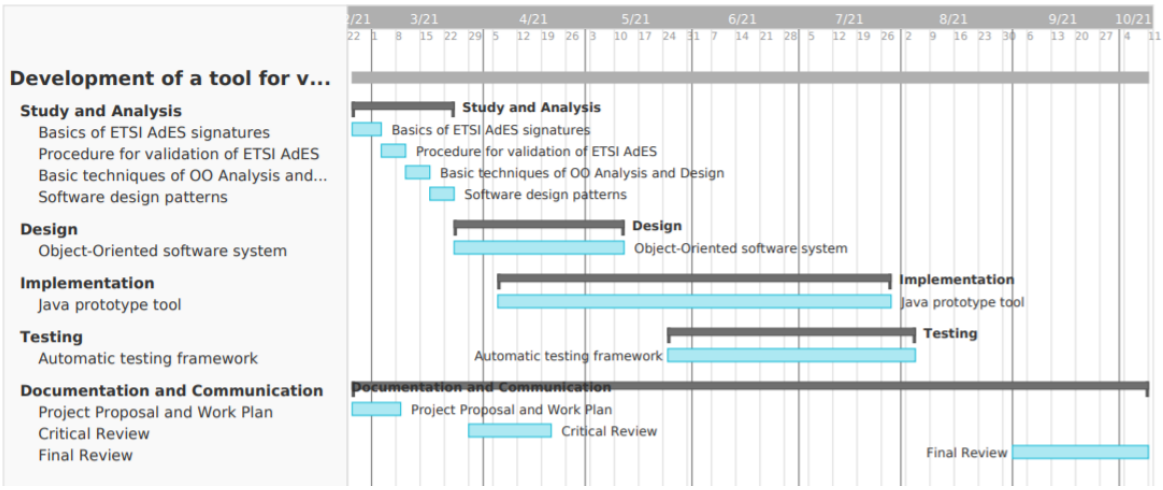

Figure 2: Gantt Diagram

1.5. Deviations and incidences

The initially proposed work methodology was cascade development. This methodology consists of a sequential process, where each step has a certain time and goes before and after others, in such a way that the beginning of each stage must wait for the completion of the previous stage. This model has a great disadvantage, and that is that any design error detected in later stages necessarily leads to redesign and new programming of the affected code, increasing development costs.

Finally, it was changed to an agile methodology, based on iterative and incremental development, where the requirements and solutions evolve over time according to the need of the project.

2. State of the art of the technology used or applied in this thesis:

2.1. What is an electronic signature?

An 'electronic signature' is a legal concept that is defined in eIDAS by the following:

"'*electronic signature' means data in electronic form which is attached to or logically associated with other data in electronic form and which is used by the signatory to sign;*" (eIDAS Article 3.10).

It is commonly confused with the concept of digital signature, which could be defined as a specific technical implementation of an electronic signature where cryptographic algorithms are applied. That is, a set of characters that serve to certify or show validity and security. They are therefore used to identify the signer and to certify the veracity that the document has not been modified with respect to the original.

The digital signature is based on public key cryptography systems that satisfy the definition requirements of an advanced electronic signature. Their operation is specifically based on applying a mathematical algorithm to the content of a document and then applying signature's algorithm, in which a private key is used, to the result of the previous operation, in this way generating the signature of the electronic document.

The main difference is that although a digital signature refers to a series of cryptographic methods, the concept of electronic signature is fundamentally legal, since it confers a regulatory framework on the signature that gives it legal validity.

2.2. Electronic signature levels

The eIDAS Regulation defines three levels of electronic signature: 'simple' electronic signature, advanced electronic signature and qualified electronic signature. The requirements of each level are built on the requirements of the level below it, such that a qualified electronic signature meets the most requirements and a 'simple' electronic signature the least.

Advanced electronic signatures (AdES)

An advanced electronic signature is an electronic signature which is additionally:

- uniquely linked to and capable of identifying the signatory;
- created in a way that allows the signatory to retain control;
- linked to the document in a way that any subsequent change of the data is detectable.

The most commonly used technology able to provide these requirements relies on the use of a public-key infrastructure (PKI), which involves the use of certificates and cryptographic keys.

Qualified electronic signatures (QES)

A qualified electronic signature is an advanced electronic signature which is additionally:

- created by a qualified signature creation device (QSCD);
- and is based on a qualified certificate for electronic signatures.

2.3. AdES profiles

Advanced electronic signatures that are compliant with eIDAS may be technically implemented through the AdES Baseline Profiles that have been developed by the European Telecommunications Standards Institute (ETSI).

How this information is structured (the order of that information within the file, the labels that indicate when a field begins and when it ends, the optionality of those fields, etc.) is determined by different formats:

- **CAdES** (Advanced CMS). It is the evolution of the first standardized signature format. It is suitable for signing large files, especially if the signature contains the original document because it optimizes the information space. After signing, it will not be possible to see the signed information, because the information is saved in binary form.
- **XAdES** (Advanced XML). The result is an XML text file, a text format very similar to HTML that uses tags. The documents obtained are usually larger than in the case of CAdES, so it is not suitable when the original file is very large.
- **PAdES** (Advanced PDF). This is the most suitable format when the original document is a pdf. The recipient of the signature can easily verify the signature and the signed document. With the previous formats this is not possible if external tools are not used.
- **ASiC** (Associated Signature Containers). Specifies the use of container structures to bind together one or more signed objects with either advanced electronic signatures or time-stamp tokens into one single digital (zip) container.

2.4. Electronic certificate

In public key cryptography, keys work in pairs, consisting of the private key, which is kept by the owner of the pair for signing, and the public key, which is made available to anyone who needs to validate a digital signature generated with the private key. The main characteristic of these pairs of keys is that what is encrypted with one key of the pair can only be decrypted with the other key of the pair.

In order to securely make available the public key, it is enclosed in an electronic document called electronic certificate. A certificate binds an identity to a public key. This binding is attested by an authority who is trusted by the users of that certificate, the Certification Authority.

In the context of the validation of the digital signature the certificate attests two things, namely: that the signature has been generated by the private key associated to the public key that is present within the certificate, and that the owner of that private key has the identity that appears within the certificate.

A certificate therefore contains the information necessary to support the act of signing, as the public key, the identity of the owner of the associated private key, signature algorithm, expiration date and issuing body (the identity of the Certification Authority), among others.

2.5. Validation of an AdES signature

To verify a signature, it is necessary:

- Check the integrity of the signed data ensuring that they have not undergone any modification.
- Check that the status of the certificate with which it was signed was correct, that is, it was valid at the time of the operation. Also, that it was a qualified certificate for electronic signature issued by a qualified trust service provider.

The requirements for the validation of qualified electronic signatures are, in particular, described in Article 32 of the eIDAS Regulation.

A summary and non-exhaustive overview of the steps involved in the validation process for qualified electronic signature would be:

- The verification of the integrity of the data;
- The verification of the validity of the certificate;
- The verification of the qualified status of the certificate and;
- The verification of the signature was created by a qualified electronic signature creation device.

In the case of the basic electronic signature, if the certificate is expired, the signature is automatically given as invalid. So how does one know if the certificate was current or not on the date it was signed?

To answer this question, the formats contemplate the possibility of incorporating additional information into electronic signatures that guarantees the validity of a signature in the long term, once the validity period of the certificate has expired.

These formats add evidence from third parties (from certification authorities) and time certifications to the signature, which really certify what the status of the certificate was at the time of signing.

Specifically, there are different signature formats that increase the quality of the signature until obtaining a signature that can be verified in the long term (indefinitely) with full legal guarantees:

- **AdES - B**: Basic Electronic Signature, is the basic format to satisfy the requirements of the advanced electronic signature.
- **AdES - T**: Signature with a timestamp, a time stamp is added in order to place in time the moment when a document is signed and to add protection against repudiation.
- **AdES - LT**: Signature with Long Term Data, certificates and revocation data are embedded to allow verification in future even if their original source is not available.
- **AdES - LTA**: Signature with Long Term Data and Archive timestamp, allows the addition of periodic time stamps to guarantee the integrity of the archived or saved signature for future verification.

2.6. Time stamp

Time stamping is a method of proving that a data set existed before a given time and that none of this data has been modified since then.

The Time Stamp is a signature of a Time Stamping Authority (TSA), which acts as a trusted third party attesting to the existence of electronic data on a specific date and time.

Time stamping provides added value to the use of digital signatures, since the signature itself does not provide any information about the time of creation of the signature, and in the event that the signer included it, this would have been provided by one of the parties, when it is advisable for the time stamp to be provided by a trusted third party.

Resealing: Since the Time Stamp is a signature made with the electronic certificate of the Sealing Authority, when that certificate expires, the stamp and, therefore, the signature are no longer valid. Therefore, before the TSA certificate expires, it is necessary to reseal or reapply the Time Stamp to maintain the temporary validity of the signature.

3. Methodology / project development:

To successfully implement an electronic signature validation software tool, it is necessary to follow a series of steps, described below.

First, it is necessary to understand the concept of electronic signature. It is necessary to find out what types of signatures exist, how they are composed, what characteristics they have, etc. A lot of time needs to be spent understanding and being clear about this before anything is implemented, as a knowledge base is needed in this area. The result of this step is point 2: State of the art.

The next step is to develop the program. To do this, the principles of agile software development are followed. This set of practices consists of iterative and incremental development processes, where the requirements and solutions evolve over time according to the needs of the project. Each iteration of the life cycle includes planning, requirements analysis, design, coding, testing, and documentation.

3.1. Analysis and Object-Oriented Design

To be able to carry out these steps correctly it is necessary to first acquire a good knowledge of some basic techniques of Analysis and Object-Oriented Design and some software design patterns. Specifically, the so-called SOLID principles are followed using techniques that ensure:

- High cohesion and low coupling in classes, keeping the responsibilities of these strongly related and focused as well as independent from the other components of the system.
- Good management of dependencies between modules and applications, building dependency firewalls, so that dependencies do not propagate through them.
- Follow the architectural principles to define the components of the software system and its limits.

SOLID represents five basic principles of object-oriented programming and design. When these principles are applied together, it is more likely to create a system that is easy to maintain and expand over time. These principles are:

- Single responsibility principle: an object should only have a single responsibility.
- Open / closed principle: entities must be open for extension, but closed for modification.
- Liskov substitution principle: the objects of a program should be replaceable by instances of their subtypes without altering the correct operation of the program.
- Interface segregation principle: Many specific client interfaces are better than a general-purpose interface.
- Dependency inversion principle: depend on abstractions, not depend on implementations.

Once the previous knowledge has been assimilated in depth, the development of the system can begin.

3.2. Signature validation

3.2.1. Signature validation model

Following the ETSI standard, the signature validation model can be divided into two parts, as can be seen in the figure 3:

- Signature validation application (SVA)
- Driving application (DA).

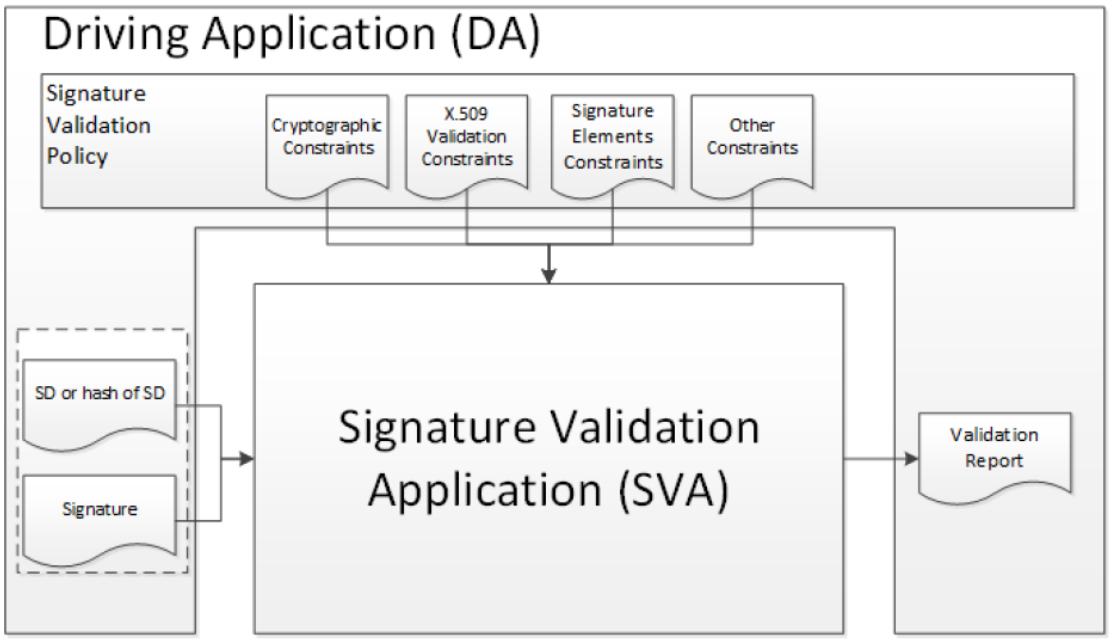

Figure 3: Conceptual Model of Signature Validation

The goal is to implement the SVA, which receives an AdES digital signature and other inputs from the DA.

The SVA shall validate the signature against a signature validation policy, consisting of a set of validation constraints, and shall output a status indication and validation report providing the details of the technical validation of each of the applicable constraints, which can be relevant for the DA in interpreting the results.

First, the data that the validation tool will receive as input must be analysed to create an optimal model to work with.

As can be seen in the figure 4, the signature structure defined in the standard consists of the signer's document, the signed attributes, which are used to calculate the value of the signature, the value of the signature itself, as well as any unsigned attributes included in the signature, the which support the signature and its interpretation and purpose.

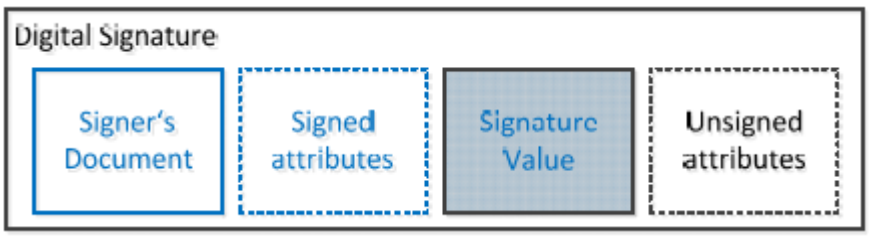

Figure 4: Digital Signature

Due to the different types of AdES signature to be validated, the proposed solution is based on the idea of abstracting the concept of signature to a general entity. For this purpose, the abstract factory design pattern must be used, which allows defining an interface to create families of related products without specifying the concrete classes of these products. In this way, the SVA will not need to know which signature family it is working with, at the same time that it will be more flexible in the

face of future changes, avoiding the coupling between classes at the moment of creation of the objects.

To model the standardized signature structure, interfaces must be used in all the objects that compose it, which will allow specifying lists of actions that must be carried out, but not their implementation, thus leaving the entity open for extension, at the same time as it will allow the validation application to only know those methods that it actually uses. In this way, it will be possible to maintain a decoupled system with respect to its dependencies, making it easier to refactor, modify and redeploy it.

Figure 17 shows the UML diagram of the created system that forms an AdES signature following the ETSI standard.

The SVA shall allow long term signature validation. It not only verifies the existence of certain data and their validity, but it also checks the temporal dependences between these elements.

The validation process follows the signature lifecycle as depicted in Figure 5 and evaluates the status of the signature based on the validation process for the first signature class of that lifecycle (Basic Signature) first. If this leads to a definitive validation conclusion (positive or negative) the validation can be stopped. However, it is possible that this signature class does not offer the information that is required to come to a definitive conclusion. In that case, the validation continues with the validation process for the next augmented signature class (Signature with Time, Signature with Long-Term Validation Material, Signature providing Long Term Availability and Integrity of Validation Material), until either a definitive conclusion is possible or no further validation process for an augmented signature class is available. The validation result of the signature validation process applied last is then the final validation result for the signature, which may remain undetermined for lack of information.

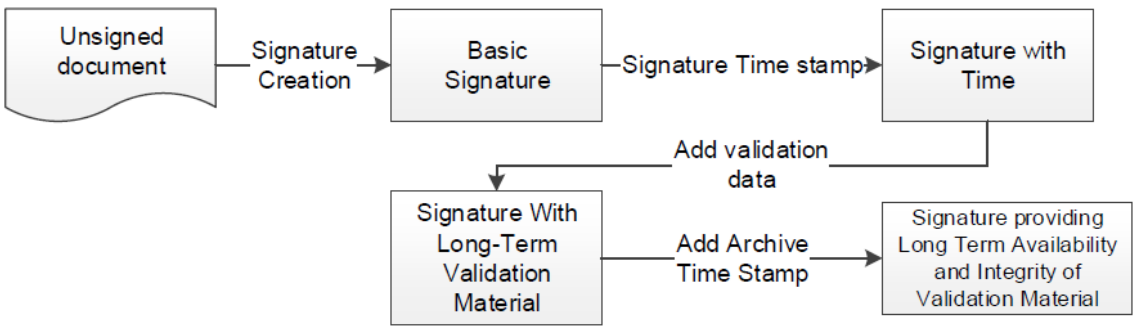

Figure 5: Signature Lifecycle

In order to conclude the validation of one of the signature classes, several validation building blocks are applied. The status on the full validation of one of the signature classes in the context of a particular signature validation policy are:

- PASSED: indicates that the signature has passed verification and it complies with the signature validation policy.
- FAILED: indicates that either the signature format is incorrect or that the digital signature value fails the verification.
- INDETERMINATE: indicates that the format and digital signature verifications have not failed but there is an insufficient information to determine if the electronic signature is valid.

For each of the validation checks, the validation process provides information justifying the reasons for the resulting status indication as a result of the check against the applicable constraints. In

addition, the ETSI standard defines a consistent and accurate way for justifying statuses under a set of sub-indications.

To characterize the SVA, the same strategy as in the AdES signature model will be followed, creating a Validator interface which will be extended through interfaces referring to each specific validation block of the signature. The implementation of each block is developed by analysing and coding the algorithm presented by the standard.

3.2.2. Basic Building Blocks

Basic building blocks are used by later clauses to construct validation algorithms for specific scenarios. The figure below shows how these building blocks are related to achieve signature validation. It closely resembles the Basic Validation.

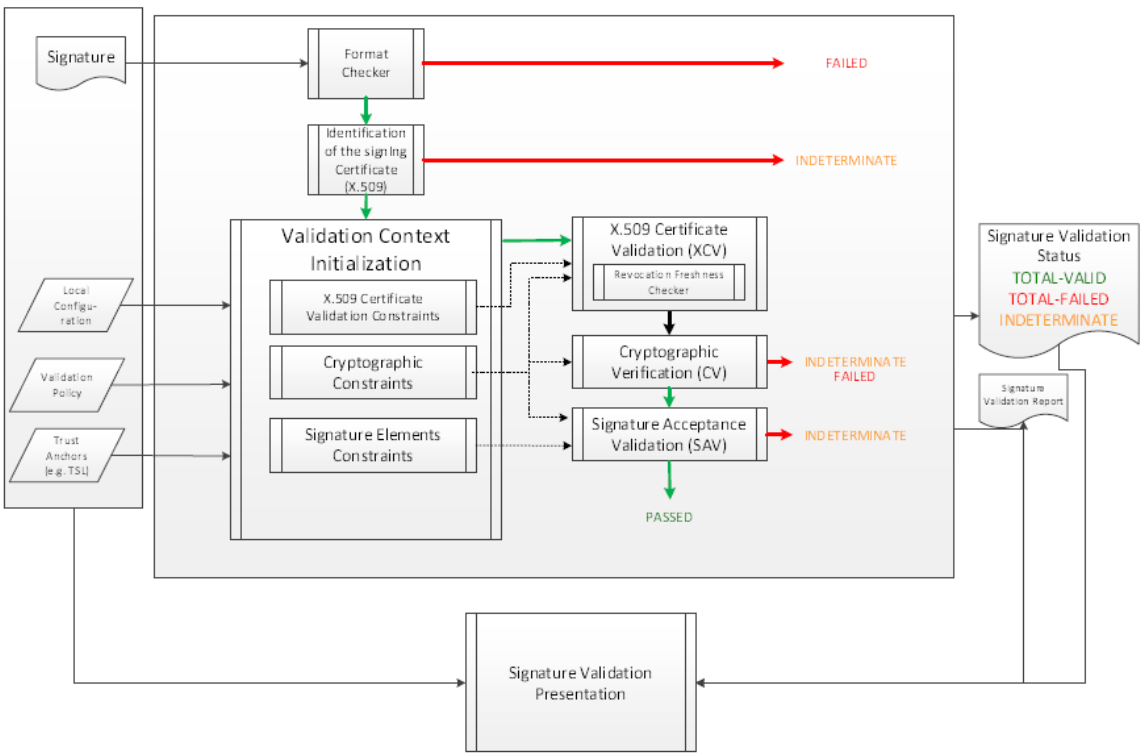

Figure 6: Basic Signature Validation

3.2.2.1. Identification of the signing certificate

This building block is responsible for identifying the signing certificate that will be used to validate the signature.

The analysis of the algorithm leads to the development of the following flow diagram:

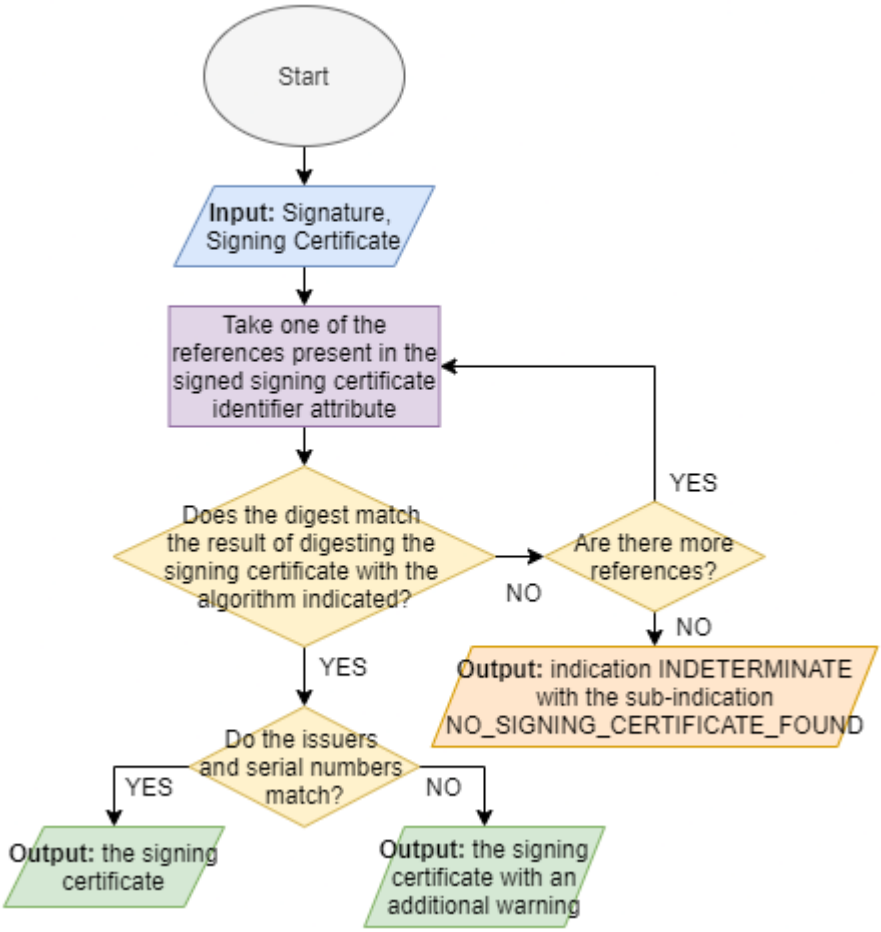

Figure 7: Identification of the signing certificate

Once the algorithm has been encoded, the block validator allows obtaining the signing certificate in case of a successful result, or, on the contrary, an indication so that together with the signature policy used, the final result of the validation can be determined.

3.2.2.2. Validation context initialization

This building block initializes the validation constraints and parameters that will be used to validate the signature.

The analysis of the algorithm leads to the development of the following flow diagram:

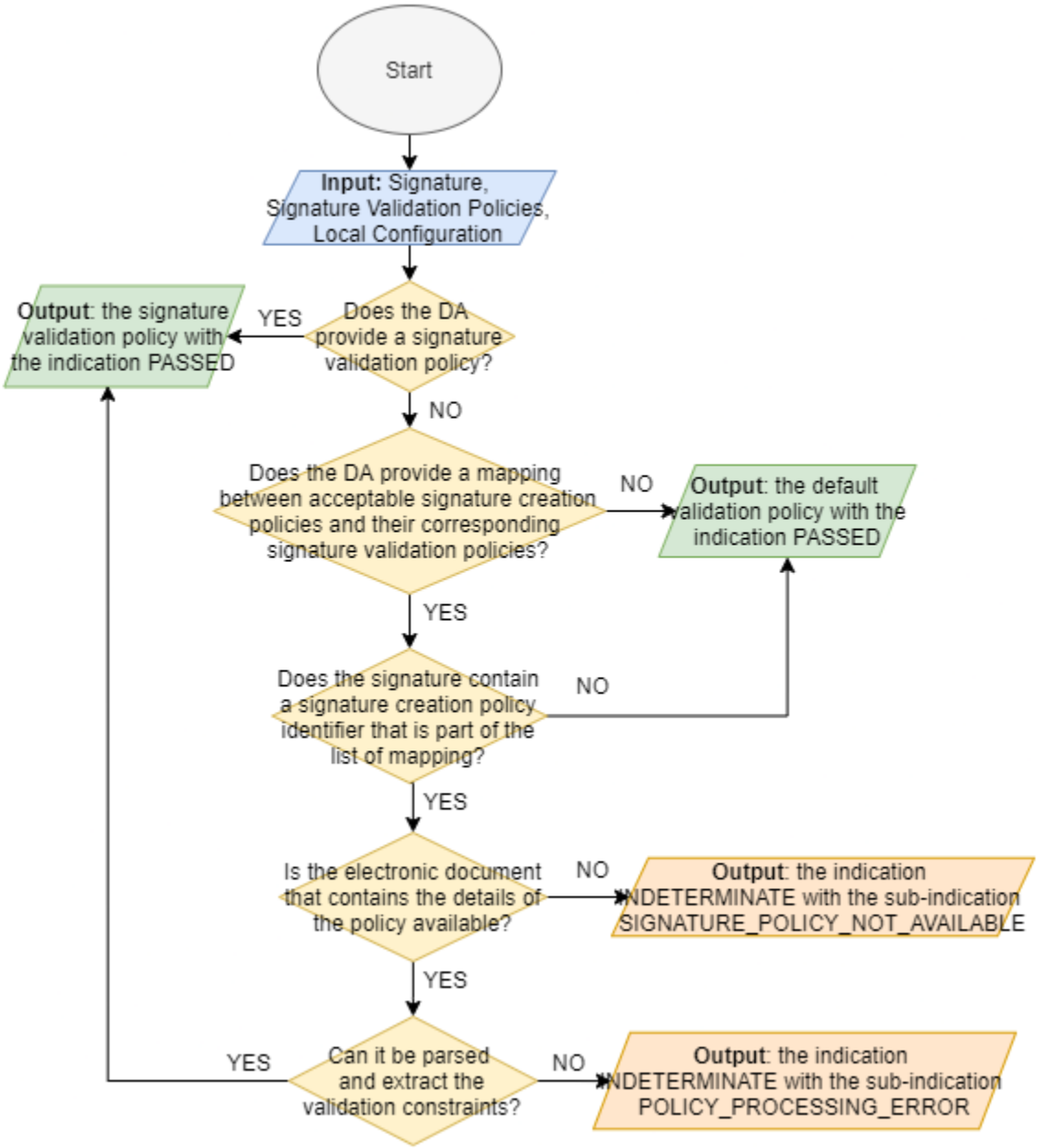

Figure 8: Validation context initialization

Once the algorithm has been encoded, in case of obtaining a satisfactory result, the block allows to initialize the X.509 validation constraints, cryptographic constraints, signature elements constraints and X.509 validation parameters, on the contrary, it returns a useful indication for the controller of validation.

3.2.2.3. Revocation freshness checker

This building block checks that a given revocation status information is "fresh" at a given validation time. The required freshness of the revocation status information is the maximum accepted difference between the validation time and the issuance time of the revocation status information.

The analysis of the algorithm leads to the development of the following flow diagram:

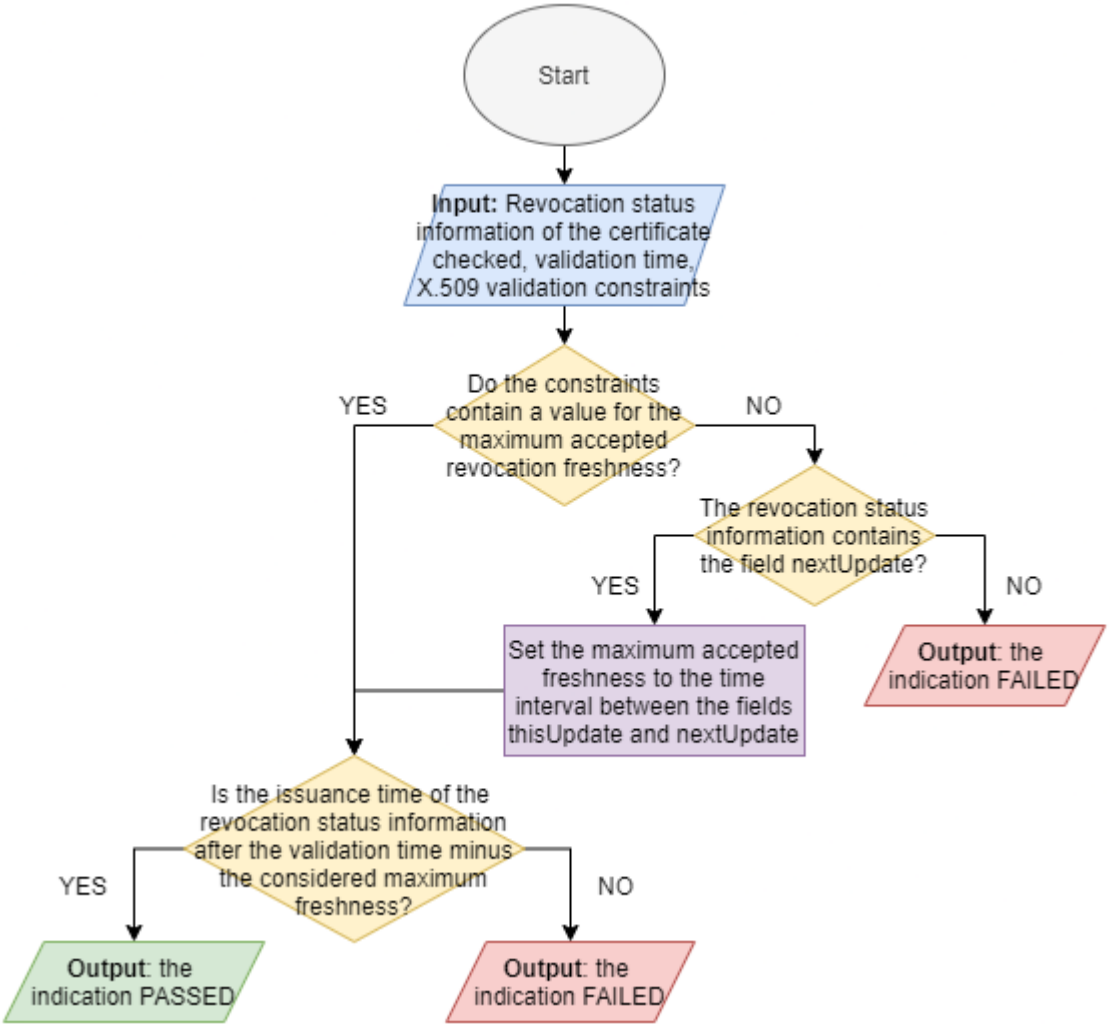

Figure 9: Revocation freshness checker

Once the algorithm has been encoded, the block allows obtaining an indication that will be used by other validation blocks when they verify the revocation status of a certificate.

3.2.2.4. X.509 certificate validation

This building block validates the signing certificate at validation time.

The analysis of the algorithm leads to the development of the following flow diagram:

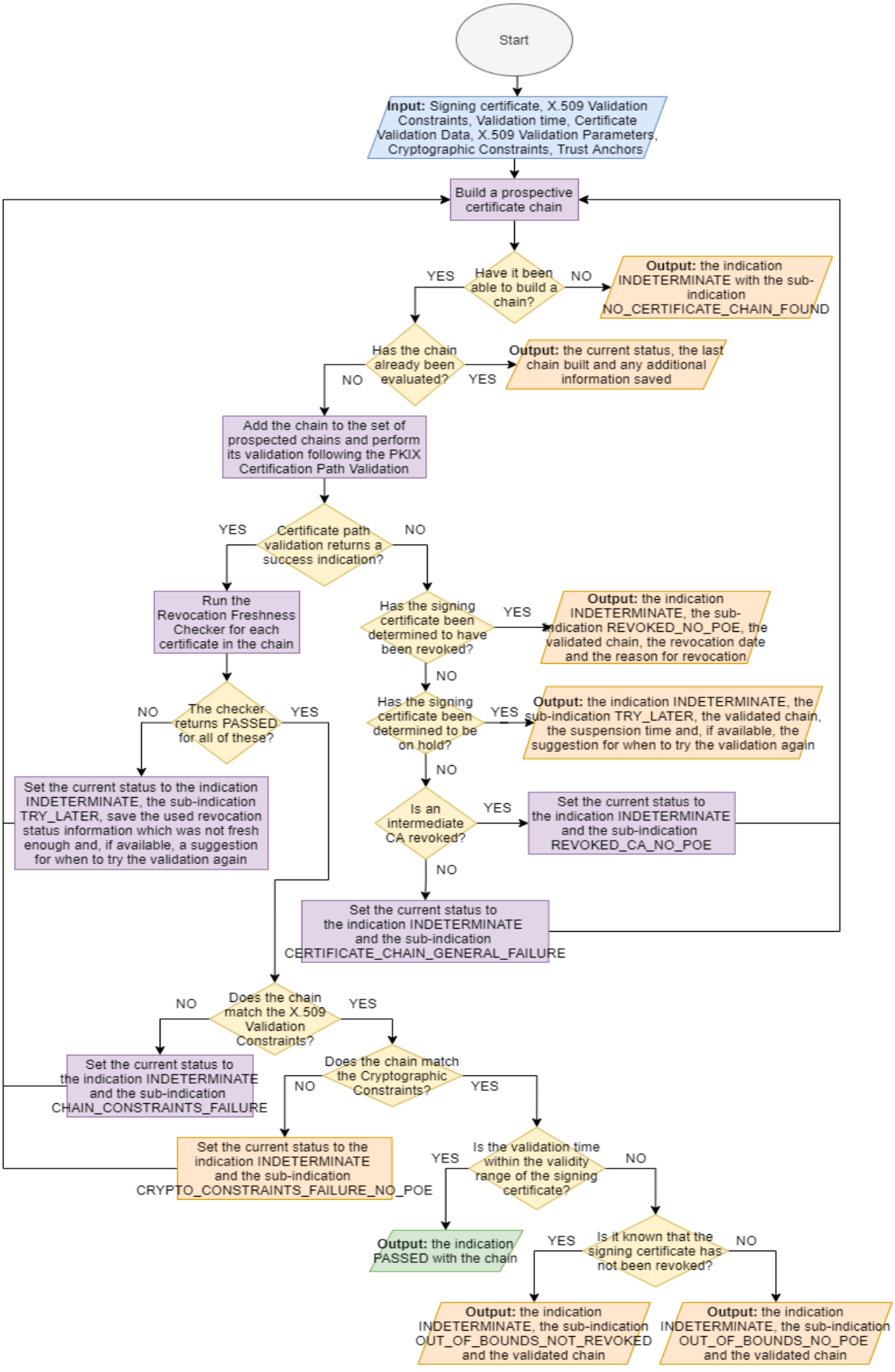

Figure 10: X.509 certificate validation

Once the algorithm has been encoded, the block, in case of successful validation, will return the certification chain used in the validation, as well as any additional validation data acquired. Otherwise, it has a set of indications that serve to know in which part of the validation process it has failed, as well as various additional information data required in subsequent basic blocks.

3.2.2.5. PKIX Certification Path Validation

This block belongs to the X.509 certificate validation building block. It implements the PKIX Certification Path Validation algorithm defined in the IETF RFC 5280 standard. The validation includes revocation checking for each certificate in the chain.

The analysis of the algorithm leads to the development of the following flow diagram:

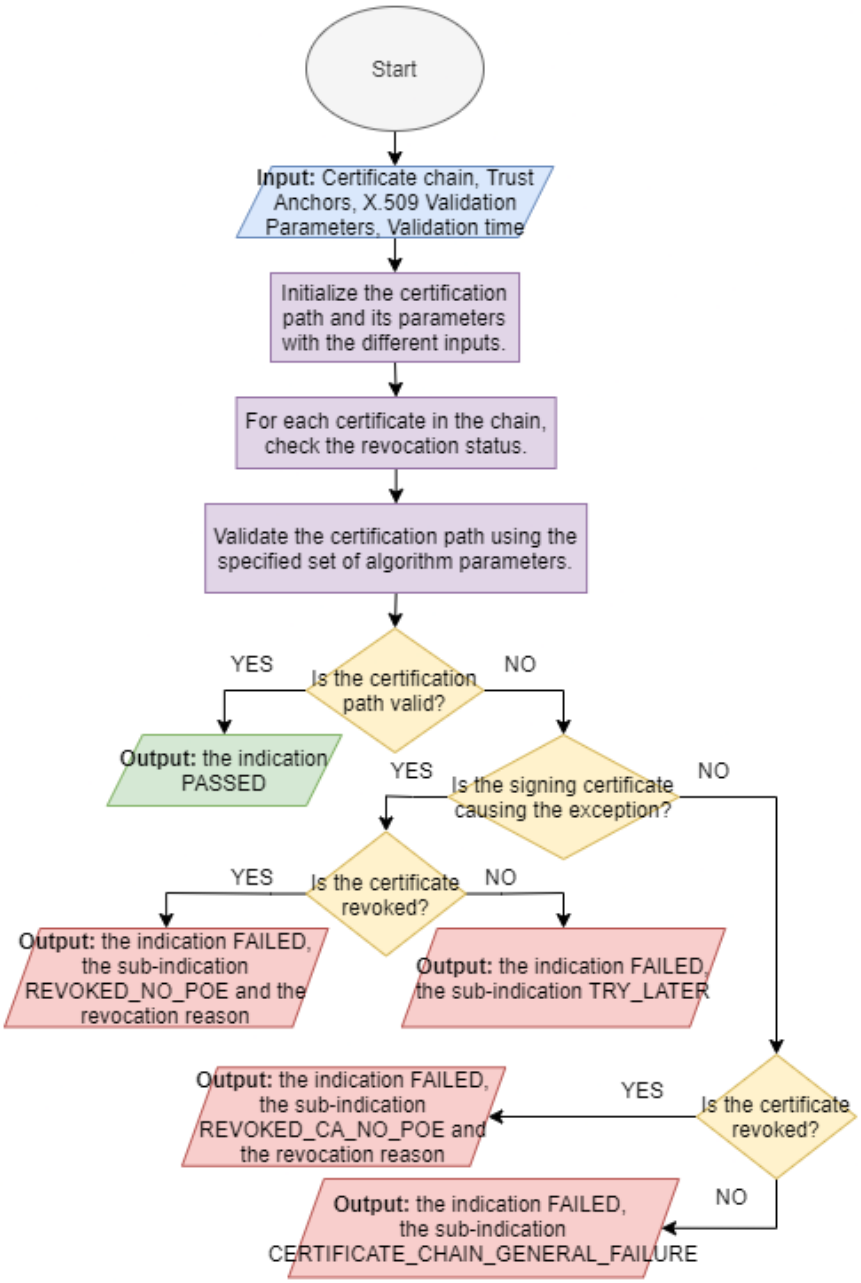

Figure 11: PKIX Certification Path Validation

Once the algorithm has been encoded, the block, in case of successfully validating the certification path, will return an indication in this regard. On the contrary, if the validation is unsuccessful due to the revocation of a certificate, it will return the wrong indication together with the respective sub-indication and the cause of revocation. If it is due to another cause, it will return the wrong indication and the corresponding sub-indication.

3.2.2.6. Cryptographic verification

This building block checks the integrity of the signed data by performing the cryptographic verifications.

The analysis of the algorithm leads to the development of the following flow diagram:

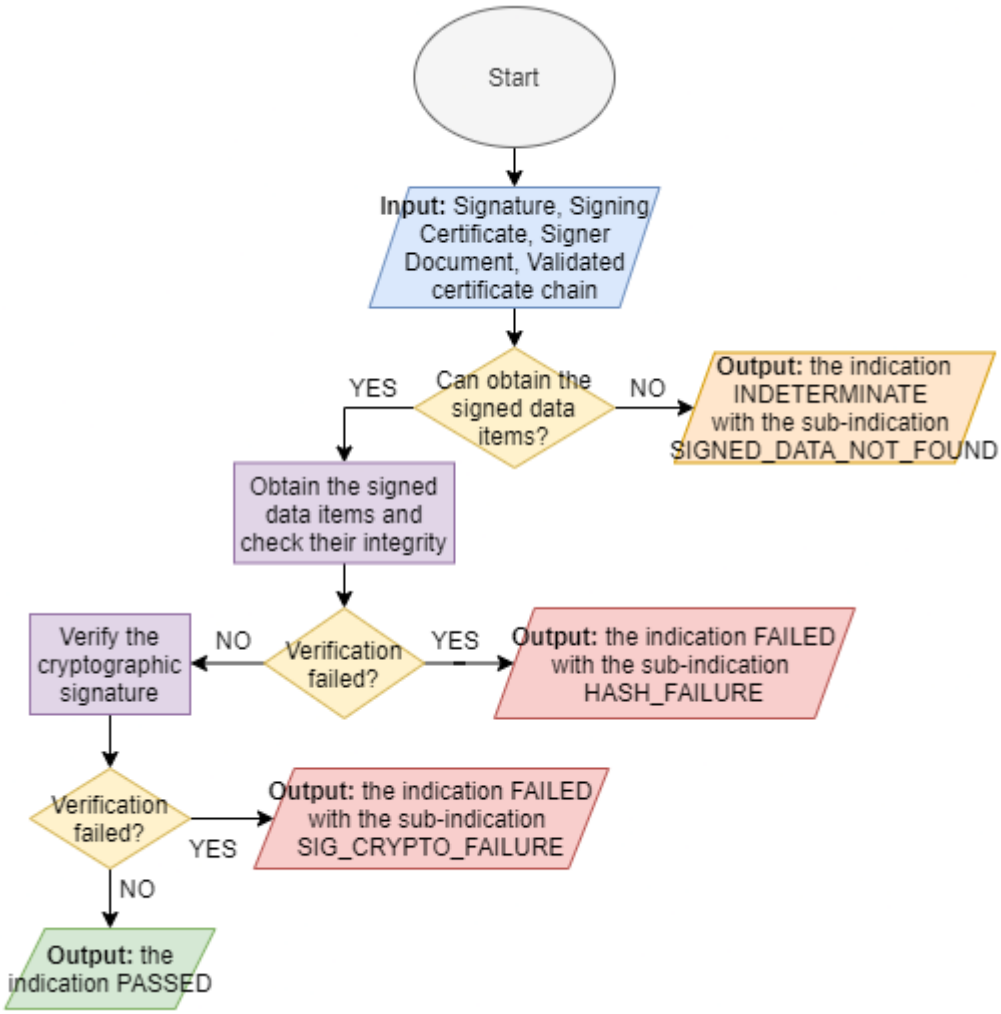

Figure 12: Cryptographic verification

Once the algorithm is encoded, the block has three possible indications as an output. In case the signature fulfils the verification satisfactorily, it will return an indication in this regard, on the contrary, in case of failure to verify the value of the signature or one of the signed data, it will return a failure indication together with a sub-indication indicating the wrong process. Finally, in case of not being able to obtain the signed data, the block will return an inconclusive indication.

3.2.2.7. Signature acceptance validation

This building block covers additional verification to be performed on the signature itself or on the attributes of the signature. This process can also include other checks mandated by a signature validation policy.

The analysis of the algorithm leads to the development of the following flow diagram:

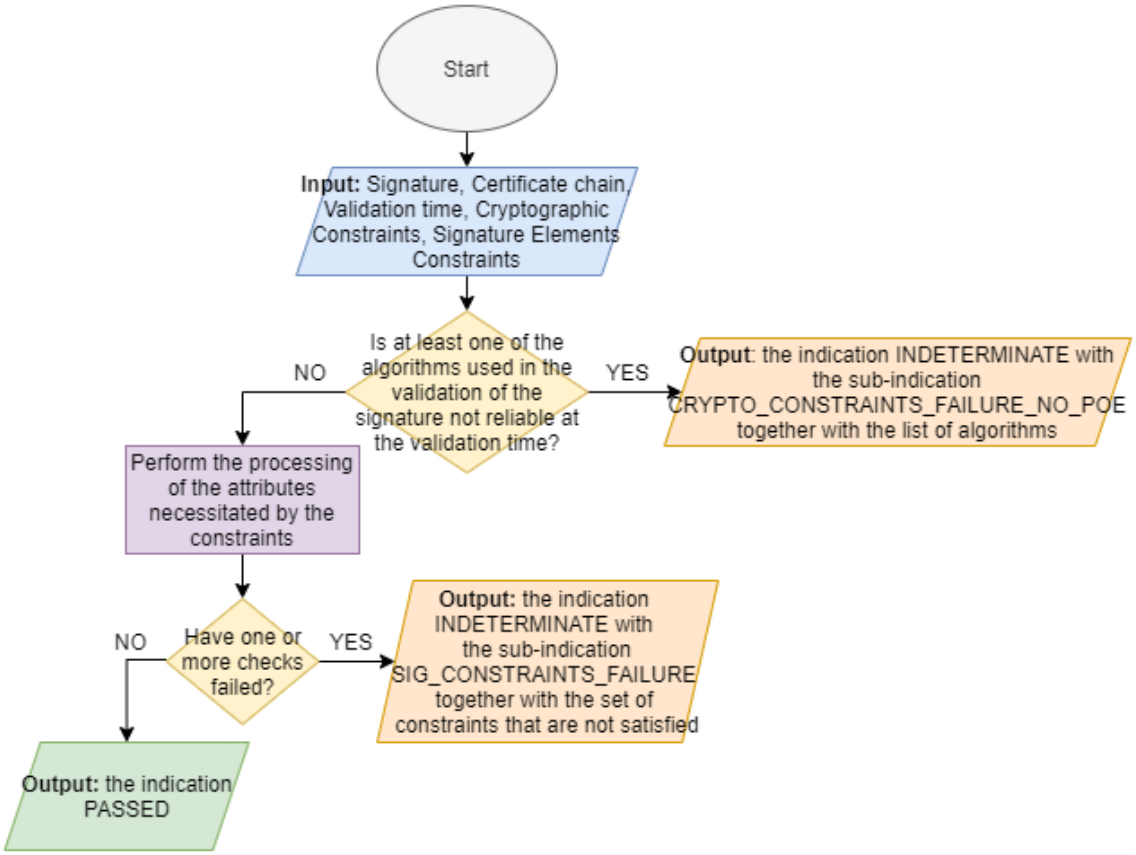

Figure 13: Signature acceptance validation

Once the algorithm has been encoded, the block, in case of successful validation, will return an indication indicating that the signature complies with the validation constraints. In case of failing any validation required by the constraints, it will return an inconclusive indication, along with the set of constraints not passed. On the other hand, if the validation failure is due to some algorithm used, it will return the indication of indeterminacy along with the erroneous algorithms.

3.2.3. Validation process for Basic Signatures

This section develops the validation process to validate Basic Signatures. This process itself is also used as a building block by the validation process of time-stamps and of Signatures with Time. The process builds on the building blocks described above.

The analysis of the algorithm leads to the development of the following flow diagram:

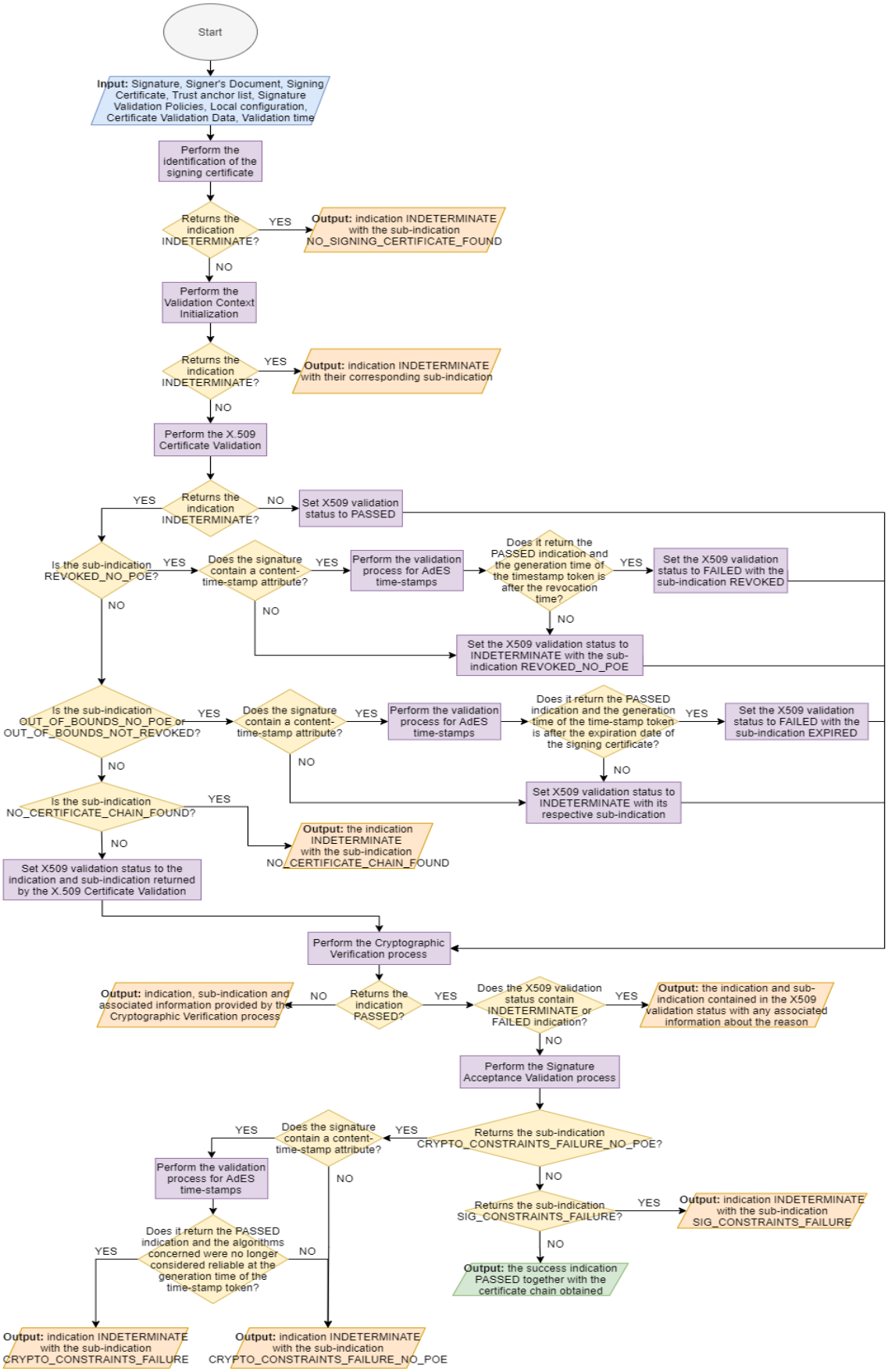

Figure 14: Validation process for Basic Signatures

The main output of the signature validation is a status indicating the validity of the signature at current time and the certificate chain used in the validation process, if applicable. This status may be accompanied by additional information.

3.2.4. Time-stamp validation building block

This building block covers the validation of a time-stamp token. According to the standard ETSI EN 319 422, a time-stamp token is a Basic Signature. Hence, the validation process builds on the validation process of a Basic Signature.

The analysis of the algorithm leads to the development of the following flow diagram:

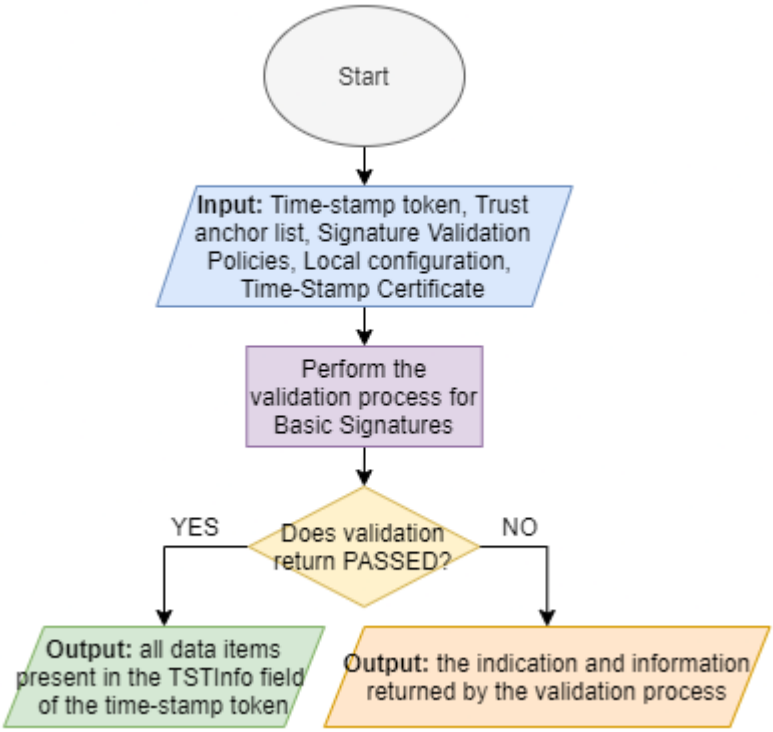

Figure 15: Time-stamp validation building block

The main output of the time-stamp validation is a status indicating the validity of the time-stamp. This status may be accompanied by additional information.

3.2.5. Validation process for Signatures with Time and Signatures with Long-Term Validation Material

This section develops the validation process for Signatures with Time and Signatures with Long-Term Validation Material. Signatures with Long-Term Validation Material differ from Signatures with Time by the fact that they contain additional validation material that can be used during validation. The validation processes are identical and are based on the process for Basic Signatures and on the Time-stamp validation building block.

The analysis of the algorithm leads to the development of the following flow diagram:

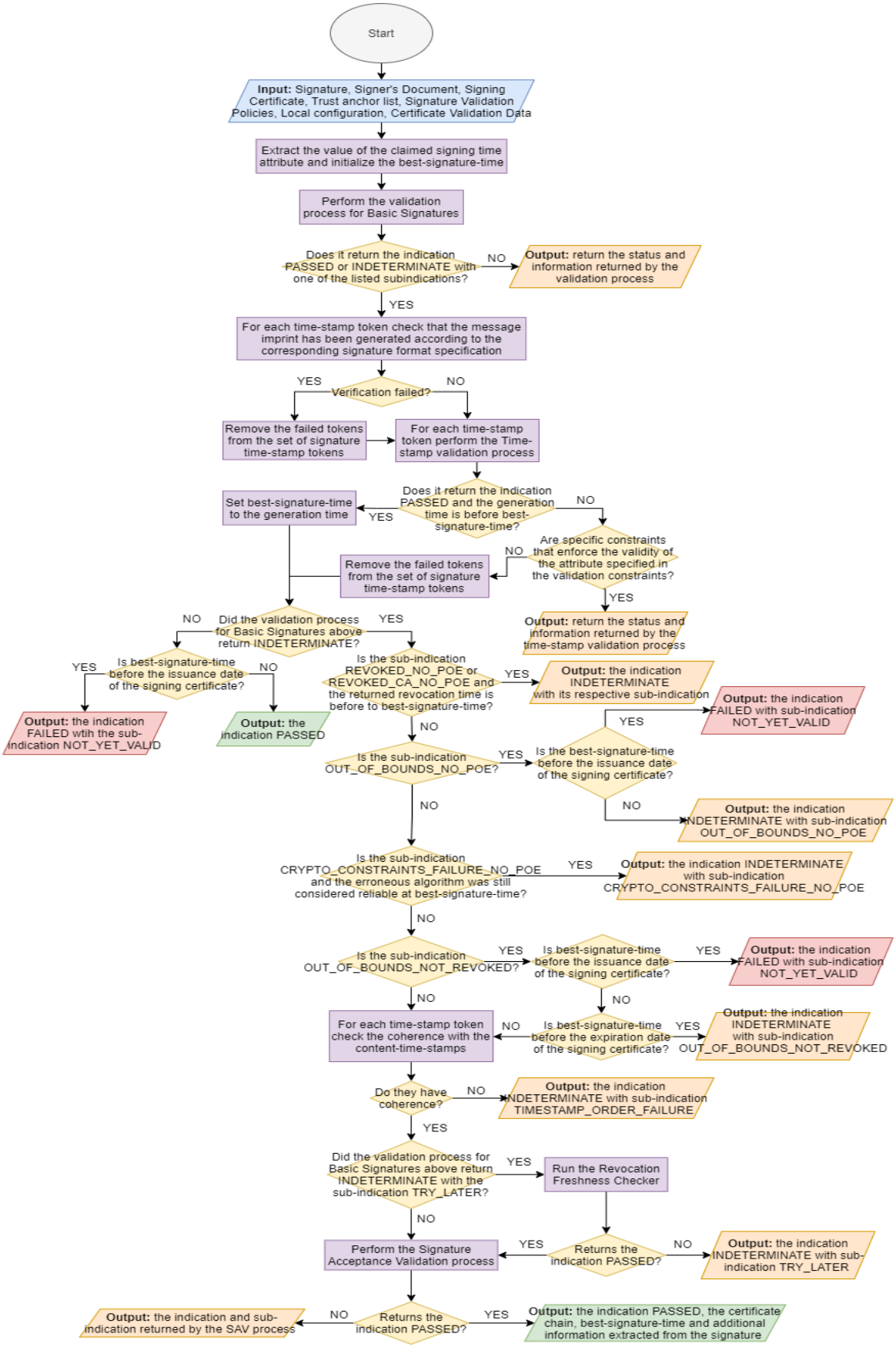

Figure 16: Validation process for Signatures with Time and Signatures with Long-Term Validation Material

The main output of the signature validation is a status indicating the validity of the signature together with the earliest time proven that the signature has existed as well as the certificate chain used for validation, if applicable. This status may be accompanied by additional information.

Finally, the figure 18 shows the UML diagram of the implementation of the set of interfaces that represent the validation system, encompassing the building blocks and signature validation processes of each scenario.

3.3. Automatic testing framework

To check the correct performance of the developed code, it must be subjected to a varied set of tests. The best way to perform these tests is by creating unit tests.

Unit tests are code snippets for testing and validating a specific functionality, behaviour, or state of the code under test. A unit test applies to small units of code, for example, a method or class. Any external dependencies must be removed or replaced by a mock object. Unit tests should not test complex user interfaces or interactions with other components.

In the case of the Signature Validation Application, these tests must be created for each validation method of the blocks that compose it, so that the correct design and implementation of the algorithm can be validated for all use cases.

Performing this set of automated tests gives reliability and quality to the project, demonstrating that the code's logic is in good condition and that it will work in all cases. Also, in the case of future refactoring of the code, it will serve to ensure that the modules continue to function correctly, and indirectly demonstrate that the SOLID principles are being followed.

For the development of the unit tests, the JUnit framework, the most popular automated test tool for Java, has been used. This provides a base structure on which to implement the tests. In addition, it provides a number of methods to check whether certain variables comply with certain characteristics, such as their values. This allows tests to be programmed and, once executed, to check whether the variables have the expected characteristics.

The Mockito framework has also been used. It is a Java library that allows to dynamically simulate the behaviour of a class. This isolates the test from dependencies with other classes and only tests the specific functionality that is desired.

The simulation of the behaviour of a class is done by creating double test objects. In the case of the application, Mock objects will be created, which are used to verify the indirect output of the tested code, first defining the expectations before the code is executed.

The following table shows the set of unit tests developed for each class in the validation system. These validate the correct operation of the algorithm developed by simulating each use case exposed in the standard.

Validation block	Type of test	Expected result
Identification of the signing certificate	Take the references and check that the digest of the certificate referenced matches the result of digesting the signing certificate with the algorithm indicated. The issuer presents in that reference matches with the issuer serial of the signing certificate.	The signing certificate.
	Take the references and check that the digest of the certificate referenced matches the result of digesting the signing certificate with the algorithm indicated. The issuer present in that reference does not match with the issuer serial of the signing certificate.	The signing certificate with an additional warning.
	Take the references and check that the digest of the certificate referenced does not match with the result of digesting the signing certificate with the algorithm indicated, until all elements have been checked. The validation of this property is taken as failed.	INDETERMINATE. NO SIGNING CERTIFICATE FOUND.
Validation context initialization	The DA provides the SVA with a signature validation policy to be used.	The signature validation policy. PASSED.
	The signature contains a signature creation policy identifier that is not contained in the list of mappings. Applies default validation constraints.	The default validation policy. PASSED.
	No signature creation policy is contained in the signature. Applies default validation constraints.	The default validation policy. PASSED.
	The signature contains one signature creation policy identifier, which is part of the list of mappings. Electronic document is not available.	INDETERMINATE. SIGNATURE POLICY NOT AVAILABLE.
	The signature contains one signature creation policy identifier, which is part of the list of mappings. Electronic document cannot be parsed.	INDETERMINATE. POLICY PROCESSING ERROR.
	The signature contains one signature creation policy identifier, which is part of the list of	The signature validation policy. PASSED.

	mappings. Extract the validation constraints from the rules encoded in the validation policy.	
Revocation freshness checker	The constraints contain a value for the maximum accepted revocation freshness. The issuance time of the revocation status information is after the validation time minus the considered maximum freshness.	PASSED.
	The constraints have not contained a value for the maximum accepted revocation freshness and nextUpdate field is not set.	FAILED.
	The issuance time of the revocation status information is before the validation time minus the considered maximum freshness.	FAILED.
X.509 certificate validation	No chain has been built.	INDETERMINATE. NO CERTIFICATE CHAIN FOUND.
	Build a new prospective certificate chain that has not yet been evaluated and add this chain to the set of prospected chains. Perform validation of the prospective certificate chain which returns a success indication. Run the Revocation Freshness Checker for each certificate in the chain. The checker returns PASSED. The chain matches with X.509 Validation Constraints and Cryptographic Constrains. Check that the validation time is in the validity range of the signing certificate.	The validated certificate chain. PASSED.
	The certificate path validation returns a failure indication because the signing certificate has been determined to be revoked.	INDETERMINATE. REVOKED NO POE.
	The certificate path validation returns a failure indication because the singing certificate has been determined to be on hold.	INDETERMINATE. TRY LATER.
	The certificate path validation returns a failure indication because an intermediate CA is revoked.	INDETERMINATE. REVOKED CA NO POE.
	The certificate path validation returns a failure indication with any other reason.	INDETERMINATE. CERTIFICATE CHAIN GENERAL FAILURE.

	The chain does not match with the X.509 Validation Constraints.	INDETERMINATE. CHAIN CONSTRAINTS FAILURE.
	The chain does not match with the Cryptographic Constraints.	INDETERMINATE. CRYPTO CONSTRAINTS FAILURE NO POE.
	The validation time is not in the validity range of the singing certificate and the signing certificate is known not having been revoked.	INDETERMINATE. OUT OF BOUNDS NOT REVOKED.
Cryptographic verification	Obtain the signed data items not provided in the input. Check the integrity of the signed data items and verify the cryptographic signature. Outputs a success indication.	PASSED.
	The signed data items cannot be obtained.	INDETERMINATE. SIGNED DATA NOT FOUND.
	Check the integrity of the signed data items and verify the cryptographic signature. Outputs a failure indication.	FAILED. SIG CRYPTO FAILURE.
	Check the integrity of the signed data items. Outputs a failure indication.	FAILED. HASH FAILURE.
Signature acceptance validation	Performs the processing of the signature attributes needed by the constraints. All the algorithms that have been used in validation of the signature and the size of the keys used are considered reliable at the validation time. All the constraints are satisfied.	PASSED.
	At least one of the algorithms that have been used in validation of the signature or the size of the keys used are not considered reliable at the validation time.	The list of algorithms not considered reliable. INDETERMINATE. CRYPTO CONSTRAINTS FAILURE NO POE.

	Performs the processing of the signature attributes needed by the constraints. One or more constraints fail.	The set of constraints that are not satisfied. INDETERMINATE. SIG CONSTRAINTS FAILURE.
Validation process for Basic Signatures	Performs a successful validation of a Basic Signature. The process returns PASSED on all building blocks.	The certificate chain obtained. PASSED.
	Cannot identify the signing certificate.	INDETERMINATE. NO SIGNING CERTIFICATE FOUND.
	The certificate path validation returns a failure indication because the signing certificate has been determined to be revoked. Additionally, performs the Cryptographic Verification, obtaining a satisfactory result.	INDETERMINATE. REVOKED NO POE.
	Obtains failure indication in Cryptographic Verification block verifying the cryptographic signature.	FAILED. SIG CRYPTO FAILURE.
	Obtains a failure indication in the Signature Acceptance Validation block processing the signature attributes. One or more constraints fail.	INDETERMINATE. SIG CONSTRAINTS FAILURE.
Time-stamp validation building block	Performs a successful validation of the token signature using the validation process for Basic Signatures.	The Time-stamp token information field.

Table 1: Set of unit tests

4. Results

Finally, after following the design and development methodology described above, the result is a software tool in Java language capable of validating Basic Signatures, Signatures with Time and Signatures with Long-Term Validation Material in XAdES format, following the European standard ETSI EN 319 102-1.

THE TOOL CONSISTS OF 82 CLASSES AND 3071 LINES OF CODE.

The design of the tool, based on SOLID principles, allows scalability, code reuse and maintenance, accepting to be extended with new functionalities in a simple and easily modifiable way in the face of future changes in the requirements of the standard.

The tool is composed of three distinct parts that generate the three necessary ingredients for the validation of a signature. These are:

- AdES signature: characterizes the ETSI AdES signature model. It includes all the parts of a signature required by validators, such as the signer's document, the signature value, and the signature attributes. Provides a set of methods to obtain the necessary data from the signature.
- Public Key Infrastructure (PKI): it characterizes the functionalities of a PKI necessary for the validation of electronic certificates. Provides a set of methods to obtain the revocation values of the certificates under study. Allows to obtain Certificate Revocation Lists (CRLs) and use the Online Certificate Status Protocol (OCSP).
- Validators: independently characterize the set of tasks to be carried out in the validation of a signature and its different scenarios. It implements the algorithm defined in the standard studied.

Likewise, it includes an automatic test package which allows to verify its correct performance through an extensive set of unit tests that simulate the different use cases in the validation of a signature. This set of automatic unit tests is really useful for the validation of the logic of the algorithm implemented when expanding the functionalities of the tool, being able to verify that it continues to work in all cases.

THE SET OF UNIT TESTS CONSISTS OF 9 CLASSES AND 2193 LINES OF CODE.

4.1. XAdES basic signature validation example

The validation of a Basic Signature is shown below together with all the results of each validation block.

Figure 20 shows an example of a basic XAdES signature, made with the dummy certificate in figure 21. From this signature example, the following data are extracted:

- SignatureMethod Algorithm.
- SignatureValue.
- SigningCertificateV2, which includes the properties DigestMethod Algorithm, DigestValue and IssuerSerialV2.

Once the above data is extracted, it is passed as input to the Validation process for Basic Signatures, together with the Signer's Document, which is shown in Figure 19. Due to the use of a dummy certificate, the X.509 Certificate Validation block of the validation process has been simulated, in order to obtain a valid certification chain.

Running the tool, with a default signature validation policy also simulated, the following results are obtained for each validation block:

Running SigningCertificateValidatorImplTest
TEST 1: Take the references and check that the digest of the certificate referenced matches the result of digesting the signing certificate with the algorithm indicated. The issuer presents in that reference matches with the issuer serial of the signing certificate. (SigningCertificateValidatorImpl.java:52)

Running ValidationContextInitializationImplTest
TEST 1: The DA provides the SVA with a signature validation policy to be used. (ValidationContextInitializationImpl.java:52)
TEST 1 REPORT: PASSED.

Running RevocationFreshnessCheckerImplTest
TEST 1: The constraints contain a value for the maximum accepted revocation freshness. The issuance time of the revocation status information is after the validation time minus the considered maximum freshness. (RevocationFreshnessCheckerImpl.java:53)
TEST 1 REPORT: PASSED.

Running X509CertificateValidatorImplTest
TEST 2: Build a new prospective certificate chain that has not yet been evaluated and add this chain to the set of prospected chains. Perform validation of the prospective certificate chain which returns a success indication. Run the Revocation Freshness Checker for each certificate in the chain. The checker returns PASSED. The chain matches with X.509 Validation Constraints and Cryptographic Constrains. Check that the validation time is in the validity range of the signing certificate. (X509CertificateValidatorImpl.java:209)
TEST 2 REPORT: PASSED.

Running CryptographicVerificationImplTest
TEST 1: Obtain the signed data items not provided in the input. Check the integrity of the signed data items and verify the cryptographic signature. Outputs a success indication. (CryptographicVerificationImpl.java:59)
TEST 1 REPORT: PASSED.

Running SignatureAcceptanceValidatorImplTest
TEST 1: Performs the processing of the signature attributes needed by the constraints. All the algorithms that have been used in validation of the signature and the size of the keys used are considered reliable at the validation time. All the constraints are satisfied. (SignatureAcceptanceValidatorImpl.java:66)
TEST 1 REPORT: PASSED.

Running BasicSignatureValidatorImplTest
TEST 1: Performs a successful validation of a Basic Signature. The process returns PASSED on all building blocks. (BasicSignatureValidatorImpl.java:210)
TEST 1 REPORT: PASSED.

5. Budget

Below is an approximation of the total hours dedicated to each task along with the total cost of the project. This calculation has been made from a part-time job of a programmer analyst, who would carry out the entire life cycle of the product, that is, the study, analysis, design, development, testing and documentation tasks. The gross hourly salary of the worker has been chosen from the internship agreement made with the university.

Concept	Amount
Study and Analysis	
Basics of ETSI AdES signatures	25h
Procedure for validation of ETSI AdES	30h
Basic techniques of OO Analysis and Design	15h
Software design patterns	10h
Design	
Object-Oriented software system	140h
Implementation	
Java prototype tool	280h
Testing	
Automatic testing framework	160h
Documentation and Communication	
Drafting of documents	80h
Total hours	740h
Gross hourly wage	9 €/hour
Gross salary	6660 €
Social charges	2130 €
Total cost	8790 €

Table 2: Budget

Keep in mind that all the tools used are free and open-source software, so there is no need to pay for the use of licenses.

6. Conclusions and future development:

The intention of this project has ranged from introducing the basic elements involved in the field of electronic signatures, such as electronic certificates or time stamps, to going deeper into more advanced concepts such as the standardisation of the creation and validation of advanced electronic signatures or the concept of the sliding window in the resealing of a signature in order to maintain its validity over time.

To carry out the project, firstly, a study of ETSI AdES signatures was carried out, in particular the XAdES format, using the ETSI EN 319 132-1 standard, which has allowed a good knowledge of the structure and characteristics of this signature to be acquired. The ETSI EN 319 102-1 standard was then analysed in depth, specifically the signature validation procedure. This has made it possible to understand the algorithm subsequently implemented.

Once the previous knowledge was acquired, different basic techniques of Object-Oriented Analysis and Design were studied, as well as software design patterns and the testing frameworks JUnit and Mockito. In this way, it has been possible to achieve a good knowledge to be able to design the software system that constitutes the validation tool.

The final result obtained, after the study, design and implementation of the standard and its algorithm, is a Java language tool capable of validating three different levels of XAdES signature. These are Basic Signatures, Signatures with Time and Signatures with Long-Term Validation Material. Together with the validation tool, a set of unit tests has been created that represent a critical element for the correct maintenance of the tool and the verification of the algorithm flow.

Finally, compiling the objectives established at the beginning, the acquisition of each of them is verified, which leads to a high degree of satisfaction in the realisation and completion of this project.

By way of future development, the implementation of the algorithm for signatures that provide Long-Term Availability and Integrity of the Validation Material can be established. As well as the different modules external to the algorithm, such as the signature validation policy analyser or the certificate chain builder. This would complete the XAdES signature validation tool, with all its use cases included.

As a further development, the creation of adapters of the tool for PAdES, CAdES or JAdES signatures would make it a complete tool capable of validating any ETSI AdES signature.

Bibliography:

[1] *Electronic Signatures and Infrastructures (ESI); Procedures for Creation and Validation of AdES Digital Signatures; Part 1: Creation and Validation.* ETSI EN 319 102-1 V1.1.1 (2016-05)

[2] *Electronic Signatures and Infrastructures (ESI); Procedures for Creation and Validation of AdES Digital Signatures; Part 1: Creation and Validation.* ETSI TS 119 102-1 V1.2.1 (2018-08)

[3] *Electronic Signatures and Infrastructures (ESI); XAdES digital signatures; Part 1: Building blocks and XAdES baseline signatures.* ETSI EN 319 132-1 V1.1.1 (2016-04)

[4] *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.* IETF RFC 5280

[5] Robert C. Martin: "Clean Architecture. A Craftsman's Guide to Software Structure and Design". Prentice Hall.

[6] Larman, Graig." Applying UML and Patterns". Third edition. Prentice Hall PTR.

[7] Portal Firma – Ciudadanos. https://firmaelectronica.gob.es/

[8] Digital Signature Service. https://dss.nowina.lu/doc/dss-documentation.pdf

[9] eSignature FAQ - European Commission. https://ec.europa.eu/cefdigital/wiki/display /CEFDIGITAL/eSignature+FAQ

[10] Mockito framework site. https://site.mockito.org/

[11] Junit 5. https://junit.org/junit5/

Appendices:

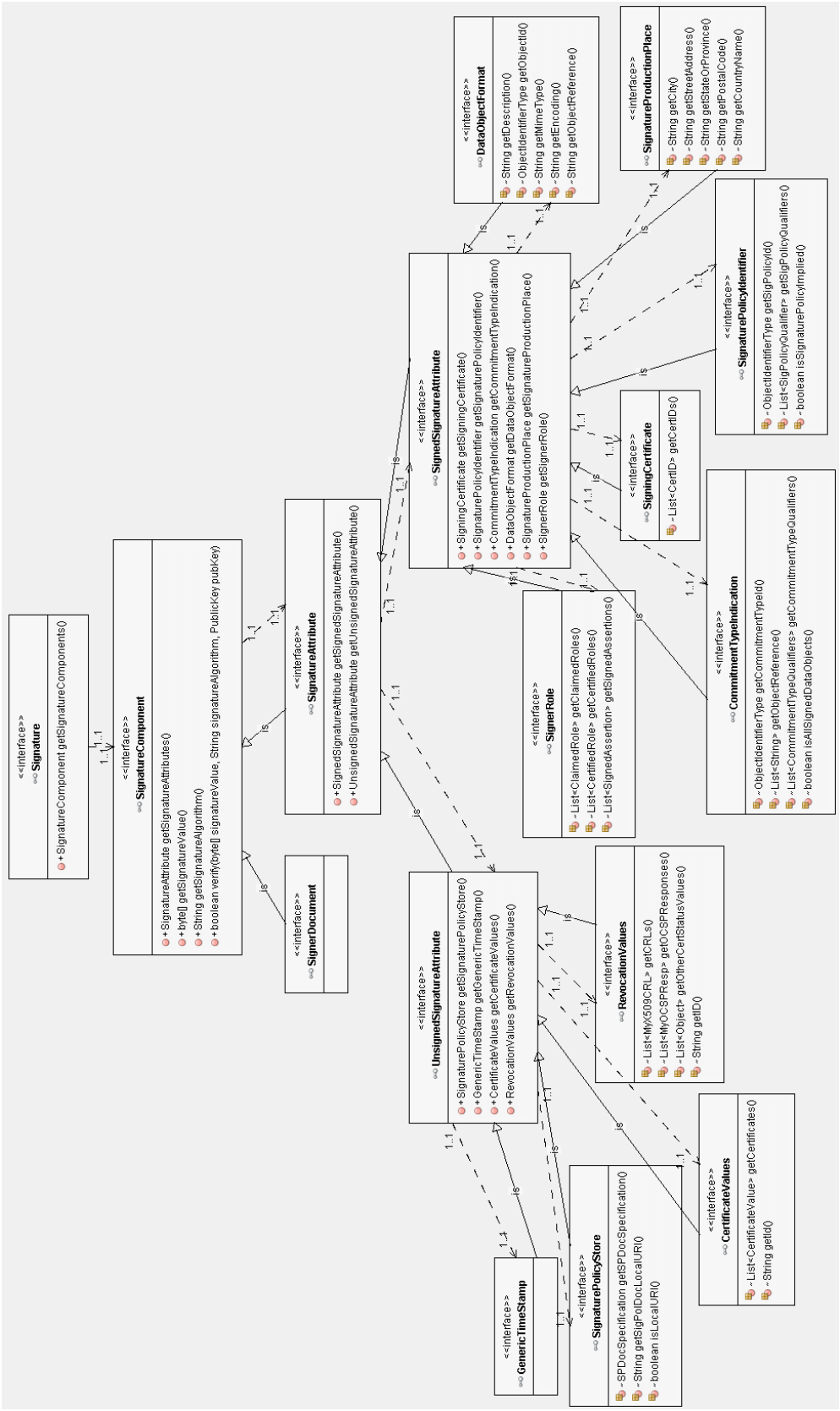

Figure 17: UML diagram of an AdES signature

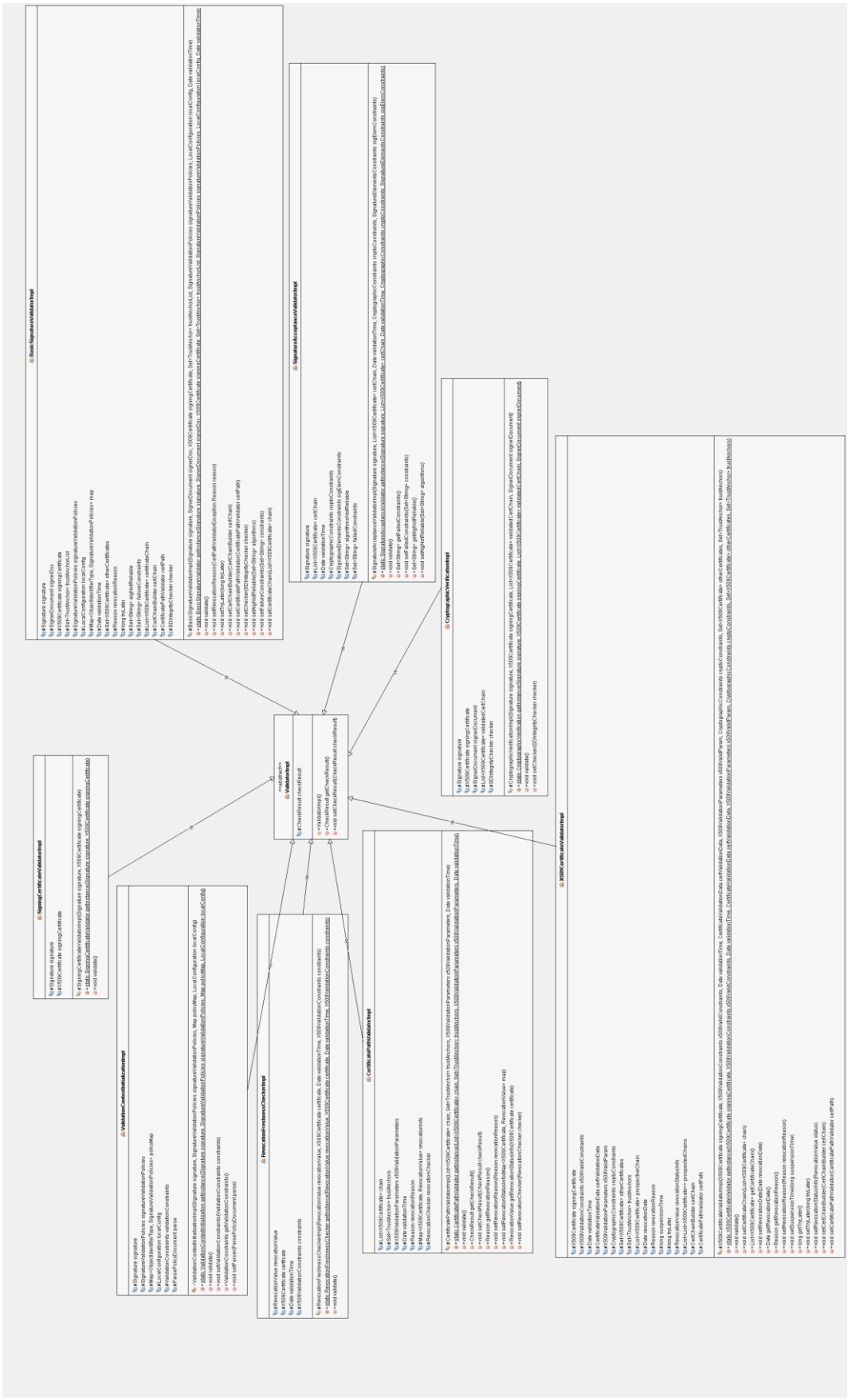

Figure 18: UML diagram of the validation system

```
<?xml version="1.0" encoding="UTF-8"?>
<text>
  <para>hello world</para>
</text>
```

Figure 19: Signer's Document example

```
<text>
  <para>hello world</para>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="id-
9bca79b1860d5334a0b3e6e9be977983">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
      <ds:Reference Id="r-id-9bca79b1860d5334a0b3e6e9be977983-1" URI="">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2002/06/xmldsig-filter2">
            <dsig-filter2:XPath xmlns:dsig-filter2="http://www.w3.org/2002/06/xmldsig-filter2"
Filter="subtract">/descendant::ds:Signature</dsig-filter2:XPath>
          </ds:Transform>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>

<ds:DigestValue>J4OsvyhJWLEW7XAEO5hPPKZGvhOFOd6wqrx6iJYAogg=</ds:DigestValue
>
      </ds:Reference>
      <ds:Reference Type="http://uri.etsi.org/01903#SignedProperties" URI="#xades-id-
9bca79b1860d5334a0b3e6e9be977983">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <ds:DigestValue>aPOlEJrMuJidgFQn/s1Ix1qHPijMJHnif4LBXkXvF2k=</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue Id="value-id-
9bca79b1860d5334a0b3e6e9be977983">KmPmWRwObvN8s8zYmkHResedIaYdGB04SqunYvj
yed1qKhmlF56yGsMl4RBIPrqvHOx0WtVVJ0kkATgCQxGmeCFZq4VtTFGq5nqqfNH8j41oUK
m6hYfD9Zx9WLI5VjGgmP5Dp1EX5NgejRxxDPTGw/Zvvlj4gurI0GNK02t2op7nSpYAhLl33p/
KrJTS6zhe6AxxLCcq9UQrbcn1ig9+YW0/vhW8JsXf+D3LY3xceQ9aq+c+2AG/EK1bRZRl9/Wh
dSjwaHTG5q5sYFcAGI7e0mzx9eZJ1Qy6lNScGf51G0oLG77ukWPO2oNPObsYP2bkAeUG5ab
ugTGsBrvYATj0rA==</ds:SignatureValue>
    <ds:KeyInfo>
```

```
      <ds:X509Data>

<ds:X509Certificate>MIIDODCCAiCgAwIBAgIEYVr3lTANBgkqhkiG9w0BAQsFADBeMQsw
CQYDVQQGEwJFUzESMBAGA1UECAwJQ2F0YWx1bnlhMRIwEAYDVQQHDAlCYXJjZW
xvbmExDDAKBgNVBAoMA1VQQzEZMBcGA1UEAwwQQ2FybG9zIENvbnRyZXJhczAeFw
0yMTA5MDExMjQ2MTNaFw0yMjEwMDQxMjQ2MTNaMF4xCzAJBgNVBAYTAkVTMRIw
EAYDVQQIDAlDYXRhbHVueWExEjAQBgNVBAcMCUJhcmNlbG9uYTEMMAoGA1UECg
wDVVBDMRkwFwYDVQQDDBBDYXJsb3MgQ29udHJlcmFzMIIBIjANBgkqhkiG9w0BAQE
FAAOCAQ8AMIIBCgKCAQEAnjRhlppu4a9/6NKSiEOXunHePrLwoE71PzqbjGPRnX2aG8/W
DX8gc5K45cYPJKxdRzbqNEd+4Z88cV71g8KPbxz1RDmHdkLDb0Cuj3lFRY/yQTHmYmwby
W+2IZGxMKJQbD86Ov6KDygt1jAzD5MYOguFYz+Pfp7dgX/R1qKeuL/3r9bKHSiBUJM9dzE
kjCCTjx0vKE+1bJDLYcXdx6cCMKsCMFsWD3phyEAd2FXxbENApt04SccaHy42ICGw9F96g
xcrfkAFlDcFR8mA4Iu4GklsP36H6ksLEzomObIS2ZwEtaq6UH4DvHveJtKa9luasGQZjY0Jq4L
DgQ0uFSGdPwIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQBsDEHnSCQw5qaT5DNSuQq
s9SynLeA2r/X7xYXW0uGQclOb9pAe/hs9JUFMBL+vspKS7lIshi69RewORXxxDeESMZLjHPIl
dTgilht/ESlQ1rlSObwhUtbmzb3u8rlCXQJuDEnuatNYE7kl0PM6Ve2SLngdD5AHygip4MIPLG7
kxmH7oC/81WmTK1Kv7ouDGFk0SFVVuQegm6LnG9DqE49ELVDouP9RhnPQKPqSmA/pMc
XpkP1ag1KTo0H3XQLaEjjLMKv00gedhtKKlrQGW0U8DrLLR1vauneSAFOAffmzjB8b/Ue7o
QNARp3qctuYjeK+ZNLWbg2mdeTJTSJ+ICZN</ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
    <ds:Object>
      <xades:QualifyingProperties xmlns:xades="http://uri.etsi.org/01903/v1.3.2#" Target="#id-
9bca79b1860d5334a0b3e6e9be977983">
        <xades:SignedProperties Id="xades-id-9bca79b1860d5334a0b3e6e9be977983">
          <xades:SignedSignatureProperties>
            <xades:SigningTime>2021-10-04T12:48:26Z</xades:SigningTime>
            <xades:SigningCertificateV2>
              <xades:Cert>
                <xades:CertDigest>
                  <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>

<ds:DigestValue>AeBf/GOofuDCrBu8m2Pgig3OMdydptOVCteeffWPxx2ct2xB0Eu80h1IF1T5X
DHJvYyT5iUw+py4x160cotO2g==</ds:DigestValue>
                </xades:CertDigest>

<xades:IssuerSerialV2>MGowYqRgMF4xCzAJBgNVBAYTAkVTMRIwEAYDVQQIDAlDYX
RhbHVueWExEjAQBgNVBAcMCUJhcmNlbG9uYTEMMAoGA1UECgwDVVBDMRkwFwY
DVQQDDBBDYXJsb3MgQ29udHJlcmFzAgRhWveV</xades:IssuerSerialV2>
              </xades:Cert>
            </xades:SigningCertificateV2>
          </xades:SignedSignatureProperties>
          <xades:SignedDataObjectProperties>
            <xades:DataObjectFormat ObjectReference="#r-id-
9bca79b1860d5334a0b3e6e9be977983-1">
              <xades:MimeType>text/xml</xades:MimeType>
```

```
          </xades:DataObjectFormat>
        </xades:SignedDataObjectProperties>
      </xades:SignedProperties>
    </xades:QualifyingProperties>
  </ds:Object>
 </ds:Signature>
</text>
```

Figure 20: Basic XAdES signature example

```
-----BEGIN CERTIFICATE-----
MIIDODCCAiCgAwIBAgIEYVr3lTANBgkqhkiG9w0BAQsFADBeMQswCQYDVQQGE
wJFUzESMBAGA1UECAwJQ2F0YWx1bnlhMRIwEAYDVQQHDAlCYXJjZWxvbmExD
DAKBgNVBAoMA1VQQzEZMBcGA1UEAwwQQ2FybG9zIENvbnRyZXJhczAeFw0yM
TA5MDExMjQ2MTNaFw0yMjEwMDQxMjQ2MTNaMF4xCzAJBgNVBAYTAkVTMRI
wEAYDVQQIDAlDYXRhbHVueWExEjAQBgNVBAcMCUJhcmNlbG9uYTEMMAoGA1
UECgwDVVBDMRkwFwYDVQQDDBBDYXJsb3MgQ29udHJlcmFzMIIBIjANBgkqhki
G9w0BAQEFAAOCAQ8AMIIBCgKCAQEAnjRhlppu4a9/6NKSiEOXunHePrLwoE71Pzq
bjGPRnX2aG8/WDX8gc5K45cYPJKxdRzbqNEd+4Z88cV71g8KPbxz1RDmHdkLDb0Cuj
3lFRY/yQTHmYmwbyW+2IZGxMKJQbD86Ov6KDygt1jAzD5MYOguFYz+Pfp7dgX/R1
qKeuL/3r9bKHSiBUJM9dzEkjCCTjx0vKE+1bJDLYcXdx6cCMKsCMFsWD3phyEAd2F
XxbENApt04SccaHy42ICGw9F96gxcrfkAFlDcFR8mA4Iu4GklsP36H6ksLEzomObIS2Zw
Etaq6UH4DvHveJtKa9luasGQZjY0Jq4LDgQ0uFSGdPwIDAQABMA0GCSqGSIb3DQEB
CwUAA4IBAQBsDEHnSCQw5qaT5DNSuUqs9SynLeA2r/X7xYXW0uGQclOb9pAe/hs9J
UFMBL+vspKS7lIshi69RewORXxxDeESMZLjHPIldTgilht/ESlQ1rlSObwhUtbmzb3u8rlC
XQJuDEnuatNYE7kl0PM6Ve2SLngdD5AHygip4MIPLG7kxmH7oC/81WmTK1Kv7ouDG
Fk0SFVVuQegm6LnG9DqE49ELVDouP9RhnPQKPqSmA/pMcXpkP1ag1KTo0H3XQLaE
jjLMKv00gedhtKKlrQGW0U8DrLLR1vauneSAFOAffmzjB8b/Ue7oQNARp3qctuYjeK+Z
NLWbg2mdeTJTSJ+ICZN
-----END CERTIFICATE-----
```

Figure 21: Dummy certificate

Glossary

ESI: Electronic Signatures and Infrastructure

ETSI: European Telecommunications Standards Institute

EN: European Standard

TS: Technical Specification

AdES: Advanced Electronic Signature

QES: Qualified Electronic Signature

eIDAS: electronic IDentification, Authentication and trust Services

PKI: Public Key Infrastructure

QSCD: Qualified Signature Creation Device

SVA: Signature Validation Application

DA: Driving Application

CRL: Certificate Revocation List

OCSP: Online Certificate Status Protocol