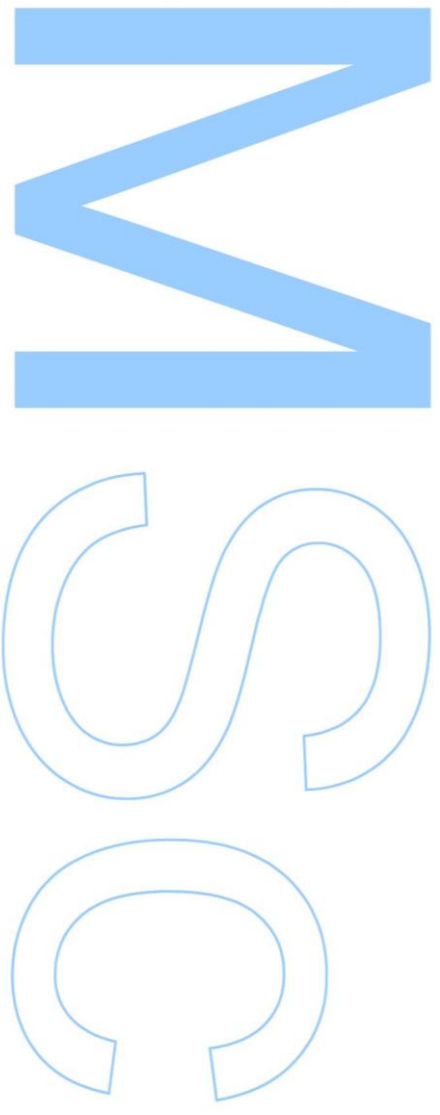
On the Integrity and Trustworthiness of web produced data

Luís A. Maia
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciência de Computadores
2013

Orientador
Professor Doutor Manuel Eduardo Carvalho Duarte Correia, Professor Auxiliar do Departamento de Computadores, Faculdade de Ciências da Universidade do Porto



U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Acknowledgments

I would like to express my appreciation for the help of my supervisor in researching and bringing different perspectives and to thank my family, for their support and dedication.

Abstract

Information Systems have been a key tool for the overall performance improvement of administrative tasks in academic institutions. While most systems intend to deliver a paperless environment to each institution it is recurrent that document integrity and accountability is still relying on traditional methods such as producing physical documents for signing and archiving. While this method delivers a non-efficient workflow and has an effective monetary cost, it is still the common method to provide a degree of integrity and accountability on the data contained in the databases of the information systems.

The evaluation of a document signature is not a straight forward process, it requires the recipient to have a copy of the signers signature for comparison and training beyond the scope of any office employee training, this leads to a serious compromise on the trustability of each document integrity and makes the verification based entirely on the trust of information origin which is not enough to provide non-repudiation to the institutions. Digitally signed documents provide an interesting solution to this problem, not only the validation of the document is automated, its integrity verifiable, but may be implemented in such way that the information contained in such documents can be directly dematerialized to different information systems without human intervention allowing cost reduction and leading to faster process workflows.

Keywords

Keywords:

- Smartcard
- HSM
- Digitally-signed documents
- Integrity
- Trust
- Long-term archives

Acronyms

HSM — Hardware Security Module

XML — Extensible Markup Language

HTML — Hypertext Markup Language

HTTP — Hypertext Protocol

IETF — Internet Engineering Task Force

IP — Internet Protocol

PKI — Public Key Infrastructure

TLS — Transport Layer Security

SSL — Secure Socket Layer

JDK — Java Development Kit

XaDeS - XML Advanced Electronic Signatures

SIGARRA

SGML - Standard Generalized Markup Language

DBMS - Database management system

SQL - Structured Query Language

EUNIS - European University Information Systems

CA - Certificate Authority

I/O - Input/Output

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The reputation of an organization is an highly important and intangible asset which plays an important role in the competitive advantage over other organizations[1]. This is also one of the key elements that is taken into consideration by the University ranking models[2], thus stimulating the institution's attractiveness to future applicants and directly influencing the final prestige of the alumni's CV.

While the institutional prestige and ranking ascertain the organization's data as a trustworthy resource,this by itself is not sufficient and some assurance about its integrity has to be provided by the institution itself in order to protect its trustworthiness and reputation.

Information produced by and circulating within the universities comes from a multitude of sources, with different trust parties, trust-delegation and information commitments.This poses an effective challenge when establishing policies that protects the interests of the institution while at the same time providing sufficient accountability mechanisms to enforce non-repudiation and commitment.

The information systems in Universities tend to be considered more complex than their counterparts in similar sized commercial organizations [3] and with the massive amount of data produced by student evaluations and student records, information assurance from record creation to long-term archive poses a serious challenge that must be properly addressed.

When establishing the required degree of information assurance [4] to manage the risks to the institutions reputation, information security plays an important key role[5]. The information system currently deployed at the University of Porto takes into account

authentication, authorization and confidentiality but lacks appropriate integrity and non-repudiation mechanisms and still relies on the inefficient document as a source of truth for its most crucial data.

After careful analysis and research of the different problems and inefficiencies of the current approach, a new architecture and process workflows providing stronger and better guarantees to information integrity across academic information systems overall is proposed.

1.1 Motivation

While the growth in complexity and functionality being provided by the University of Porto information system, centralizing every work task in the institution, namely user management, course management, professor attribution and grading, transformed the information system from an ad-hoc tool to the source truth for academic records, the existing systems still rely on the institution blind trust on the data integrity in the system databases and paper based records.

In order to reassure that a person with special privileges across the infrastructure is not able to modify or otherwise manipulate parts of the data, a document copy on paper is produced, requiring traditional physical verification of the document and of the imprinted signatures or other marks like seals or stamps. This sort of integrity checks have some costs and pose a few more challenges as to when and how the integrity is verified, how to verify forgeries and what early mitigation and fraud detection must be put into place to mitigate this situations. This materialization and dematerialization of paper based documents to assure integrity is not only error-prone and cost-ineffective due to human and supplies costs, but it also requires an extraordinary effort on the maintenance of huge volumes of paper in archives making early mitigation and fraud detection highly unlikely.

The evolution of information systems to a purely paper-less environment is the envisioned path. It would allow for the reduction of costs, a higher degree of freedom, speedier workflow and an extra degree of security by providing external entities the ability to automatically verify document integrity and easier materialization and dematerialization of documents in and out of different information systems. By providing new types of digital documents which can be trusted, a scenario where different institutions can share verifiable documents and directly import them into

their own information systems, without human intervention, would not only benefit the institution but would also provide a useful tool for information sharing across different institutions.

1.2 Objectives

Huge volumes of documents are produced by academic information systems. While we can extend our proposals to all the different kinds of documents currently generated by the system, the main focus are the documents produced during the students grading process. Due to their transitive nature and to the complexity of implied trust hierarchy which requires the delegation of trust, creating a trust-chain from the university to each faculty, and finally to the professors grading the student. By managing to retain an high level of trust on documents employed in the grading process, the ability to produce subsets for any kind of document involved in business processes that require a similar trust chain should be effortless and mainly focused on producing new document structures that can include the data that needs to be stored.

In this dissertation, we propose an entire subsystem, from the analysis of the implied structured trust hierarchies already existing in the institution to the development of the supporting infrastructure, web services, definition of work procedures, document structures and archival methods.

1.3 Features

The proposed architecture and its implementation touches on the following topics:

- *Delegating trust* - The process of entrusting an individual or a set of individuals should retain the non-repudiation property. Since this delegation of trust is not immutable with time, long-term archive of documents describing this delegations is required.

- *Revoking trust* - Trust can be removed from an individual at any time, since a delegation establish a trust time-frame, revoking trust on an individual requires an extra document stating that trust is revoked from a specific delegation document. Newer document should be able to re-establish trust on the individual.

- *Mutual Non-repudiation* - The institution which accepted the document and the signer should not be able to challenge a document. An accepted document automatically binds the document to the author and the institution which had to verify and accept that the author had the ability to produce such a document.

- *Proof of authorship* - Documents can be produced in context of a workflow pertaining to a group, the document must bind a specific author to a document created in this context.

- *Trusted-party commitment to produced information* - Every document must carry a trusted-party timestamp. The trusted party should be a third-party outside the control of both the user and institution.

- *Institutional commitment to produced information* - Documents stored in the institution's archives with valid signatures and third-party timestamps are considered irrevocably true.

- *Data integrity assurance in long-term archives* - The data contained inside documents is duly signed and archived following good practices for long-term archives. The institution is responsible for producing new signatures for documents which cannot be externally verified or no longer containing valid certificates.

- *Efficient document materialization and dematerialization* - Document's materialization should carry enough information for third parties to verify the documents are legitimate.

- *Efficient workflow* - The workflow for producing signed documents should be mostly transparent to the user and unobtrusive.

- *Policies enforcement* - Implementing a new paradigm for documents workflow requires a strong policy establishing which documents are required to be digitally signed and which documents can follow the old procedures.

- *Auditing capabilities* - Institution archives are required to be recurringly audited, and preemptively verified against fraud and should have put in place secure mechanisms to facilitate these tasks.

1.4 Proposal

This dissertation presents a new approach for the existing information system workflows currently in use by the University of Porto, which requires signed documents to be physically produced, and proposes a paradigm where paper signature requirement is naturally unneeded. In this new paradigm, it is possible to extend existing verifications and establish a trust chain between the institution and the document signee that can be easily verified by third-parties, while maintaining integrity,accountability,auditability and non-repudiation of the produced documents.

1.5 Application Domain

Dealing with data integrity and asserting trust is a huge transversal problem to information systems. Although the solution presented in this dissertation is centered on the university's information system, the same principles apply to other information systems with similar requirements. Taking into account the volatile hierarchies, trust attribution and delegation delimited by timeframe, staff autonomy and massive amounts of information being shared with other institutions implying both materialization and dematerialization of documents, the general solution devised in this paper can be applied to information systems with only a subset of these requirements.

1.6 Contributions

The research during the elaboration of our proposal lead to interesting developments and enabled the possibility to publish two papers while at the same time develop solutions to some challenging problems.

- *Smartcard support in Android*

  Android based tables are a growing trend around the university campus. Although the initial intentions of supporting this devices, not being able to provide strong non-repudiation by using smartcards and simply using a software based solution was a compromise we did not intend to make.

  MicroSD cards with secure elements were an obvious solution, but although they offered solutions based on manufacturers SDKs there was no apropriate solution

enabling portable code with simple abstraction around Java classes. Android application developers would need to write code specifically to each microSD manufacturer.

At least one of the manufacturers were at the time intending to support a PKCS#11 wrapper in their product and the huge amount of users requesting gave us the incentive to solve this problem.

The method for porting the required tools and providing a simple abstraction to the Android Java virtual machine enabling the use of Java keystores with the secure element present in the MicroSD card was published in a paper[6] at the CISTI2012 conference.

- *Document sharing format*

  With the analysis of the standard methods for sharing information across European institutions during mobility exchange programs, a new type of document was devised and a paper[7] published at the EUNIS conference.

- *XaDes Support for Signserver*

  PrimeKey's signserver[8] did not support XaDeS and due to the opensource nature of the project, we have developed a module that enables signing and validation workers for XaDeS. This functionality was highly requested in the project forums and we have therefore produced a patch that was later adopted by PrimeKey's development team and integrated into current releases.

  During the test phase of the developed module, while cross-validating the produced signed documents with a different library (*Componentes de Firma*) created by the Spanish government, a bug in the document validation[9] was identified and reported to the library developers.

- *PDF Signature Applet*

  Following the needs to improve the trust-ability of documents in exchange programs, an applet providing the ability to sign PDF documents in SIGARRA was developed.

  This Java applet provides the ability to create a visual representation of each signature, where each specific document tag is found. The first document signer, before performing the signature parses all the information system internal tags and generates the according signature fields, enabling external entities to perform easy signatures using different tools.

1.7 Outline

The remaining chapters of this thesis are organized as follows:

The description of the current state of the information system in chapter 2, not only describes the current workflow but identifies risks to data integrity and trustworthiness. Exploring the institutional hierarchies and implied trust chain provides insight into the main problem with trust delegation.

Chapter 3 contains the technical background required for understanding the architecture and design of our solution. The solution is proposed in chapter 5 along with the analysis of each problem identified and their associated solutions to mitigate the risks introduced in chapter 2. Chapter 6 focus on the implementation of the changes to the information system to support the proposed solution, with an overview of the main architecture, components and the software needed to support these changes.

In chapter 7 we explore the difficulties that arose during implementation, as well as the derived proof of concept and effectiveness of the proposed solution in fraud detection and its mitigation.

Chapter 2

Information system at the University of Porto

In this chapter we succinctly describe the current information system in production at the University of Porto and the tools that will play an important role in the development of the proposed solution. While many of these tools are already developed and in use, due to the specificity of our environment, with shifting hierarchies of trust. We also describe a new subsystem that provides the infrastructure needed to delegate trust and the trusted signee a platform to perform digital signatures that plays a crucial rule on creating an Institutionally trusted document archive that can maintain its long-term integrity and accountability even when assuming that some elements of staff can not be fully trusted.

2.1 Information System

The information system currently in use was created in 1992 and it has been under development until today. While its first version in 1992, developed by the rectorate, had the main objective of managing student records in the faculty offices, by 1996 an integrated system with web interface was developed thus setting the path to provide a web oriented information system [10].

While in the following years huge improvements and new functionalities have been developed [11], this information system was still not being used by the entire academic community, and it was not until the year 2003 that this information system, denom-

inated SIGARRA, spread to most faculties transforming itself from an important tool for academic records management into the truth source of academic achievements itself. The spread of the information system across every faculty in the University of Porto created the opportunity to develop solutions extendable to the entire university, allowing information sharing across the entire institution and providing the framework to move the entire grading workflow to a paperless environment.

Since the information system has a huge set of functionalities unrelated to our objectives,only a subset of functionalities related to the management of academic records are described in the following subsections which provide some insight on the changes required to implement grading document's signature and establishing their allowed signees.

2.1.1 Database

The U.PORTO Information System is supported by an Oracle SQL relational database. Data is added to the database via the application layer of the information system, the information is recorded and simultaneously logged with timestamp values and information about the user who interacted with the system. All changes to the database are recorded in the change logs of DBMS, enabling auditability mechanisms[7].

2.1.2 Document's Dematerialization

Materialization and dematerialization of documents is a very important process in the university workflow. When considering separate instances of the information system and documents moving across institutions, easy document materialization and dematerialization between different instances of the information system becomes important.

Verifying a physical document integrity is not a simple process, as it requires the verifier to have a pristine copy of the signature for comparison and trusted documents containing the university stamp to compare stamp features. This poses yet another problem, staff training to spot signature forgeries is nonexistent and a time-consuming task[12], leading to a huge effort in the verification of documents in transit between information systems or, in the worst scenario, the non-verification of their integrity.

In order to address the difficulties with information sharing across different institu-

tions participating in a student exchange programs, Europass guidelines have been drafted[13] in an attempt to standardise the document format used in such exchanges.

The Europass specification is divided into 5 different types of documents:

- *Curriculum Vitae* - Document defining a curriculum vitae structure for the presentation of an individual's qualifications and competences
- *Mobility* - Document recording study periods abroad in an exchange program.
- *Diploma Supplement* - Document supplying information on its holder's educational achievements at higher education level.
- *Certificate Supplement* - Document describing the competences and qualifications corresponding to a vocational training certificate
- *Language Passport* - Document providing individuals with the opportunity to present their language skills.

The main focus is on the three types of documents that are issued to citizens by competent organizations, namely the Europass Mobility, the Europass Certificate Supplement and the Europass Diploma [14] which has been jointly developed by the Council of Europe and UNESCO [15].

In the context of information sharing across different institutions, document materialization is as important as document dematerialization.An effective solution must enable the ability to import materialized documents to new information systems, preserving the integrity verifications and document structure as well as the human-friendly form. Document types that provide such flexibility have been presented at EUNIS[7] which can help shape the standard for document sharing among the European partners .

2.1.3 Risks to data integrity

In this current information system, the database does not adequately separate domains of trust [16] and the database administrators as well as every member of staff which has privilege in the information system servers have the power to modify and manipulate any data related to student records, effectively positioning themselves as a trusted party in the institutional trust chain and increasing responsibility and accountability

in the staff. Also, any intrusion to the information system or its databases may result in changes to the information pertaining to student academic information that would be almost blindly trusted by the institution except for a manual verification of the paper trail which, obviously, has associated costs.

When moving away from a paper-backed environment, the threat posed by a lack of the separation of trust-domains is not exclusive to whomever has the ability to manipulate the data, but the simple possibility of offering a degree of distrust in the integrity of the data allows the issuer of each grade the ability to repudiate the previously submitted information, thus subverting the required non-repudiation property on a trusted information source.

2.2 Trust

The concept of "Trust" derives mainly from the social sciences and can be defined as the degree of subjective belief about the behaviors of a particular entity[]. Trust is also context-dependent coexisting at different levels even between the same agents within a collective.

Huaizhi [17] defines trust as a measure on the belief that an entity is capable of performing reliably, dependably, and securely within a certain context. Trust management has traditionally been described as a special case of risk management and classified as evidence-based trust management and monitoring-based trust-management.

Although there is so debate about the notion of transitivity and unintentional transitivity in trust [18], under certain semantic constraints, trust can be transitive [19], given that a trust referral mechanism preserving the context in which trust given is used.

2.3 Hierarchy and trust

The University of Porto is comprised of fourteen faculties and a business school, providing a large variety of courses which covers the whole range of study areas and all levels of higher education. The schools of the University have administrative autonomy, and this organizational separation creates the need to ensure that the information systems maintain local particularities and allow administrative processes

to be dematerialized contemplating the structures defined in the University hierarchy.

In heterogeneous academic environments such as this, a technological solution must enable the transfer of University's organizational hierarchies to the digital world, ensuring for example that the documents electronically generated are digitally signed and retain the required hierarchical dependence.

When replicating such hierarchical dependence, trust has an implied transitive nature. Trust is delegated not uniquely on an individual but also for each task within a given time-frame. While tasks can be entrusted to an individual, they can also be delegated to a set of individuals that can only produce a valid document by cooperation.

Establishing and maintaining this degree of flexibility in trust chains requires deep changes into the information system and associated workflows, the definition of domains and sets of trust for each task while producing tamper-evident documents and providing non-repudiation properties to protect both sides of the trust chain. Recreating this institutional trust hierarchy which may not be known to external entities requires that all the final signed documents available to external entities must contain the signature of the most trusted signee, namely the institutional signature being provided by the digital signature services and the timestamp obtained from an external entity as to offer non-repudiation by the university itself.

2.3.1 Role attribution as delegation of trust

The production of any trusted document requires the ability to assert if the requester has been entrusted with the ability to generate a document for the given intent. While authentication and authorization play an important role on the production of documents, the policy establishes that documents must be materialized and signed. This requirement arises from the fact that authentication does not assert identity by itself thus enabling the non-repudiation of documents. When authorizing a user to produce any given set of documents such as grading terms, a legally binding document entrusting individuals this ability must be produced.

While multiple frameworks have been defined for cascade delegation[20] [21] and specifically hierarchical delegation tokens[22], in order to preserve the real-life implied hierarchies and multiple delegations with respective storage in trusted long-term archives, a new set of documents representing the trust delegation have to be created.

2.3.2 Student Evaluation

The current grading process in the information system can be described by the series of events required to produce a grading record as described in Figure 2.1.

The academic services create a term for each evaluation period, distributing that term to the professor responsible for each curricular unit. When entrusted that term, the professor proceeds with the grading of each student present in the document and submits it to the academic services again where the information would be validated and registered. After this process of validation, this document is sent to the professor back again who signs the document committing himself to its content and sends it back to the academic services for archival, at which point the student classifications in the information system are considered final.

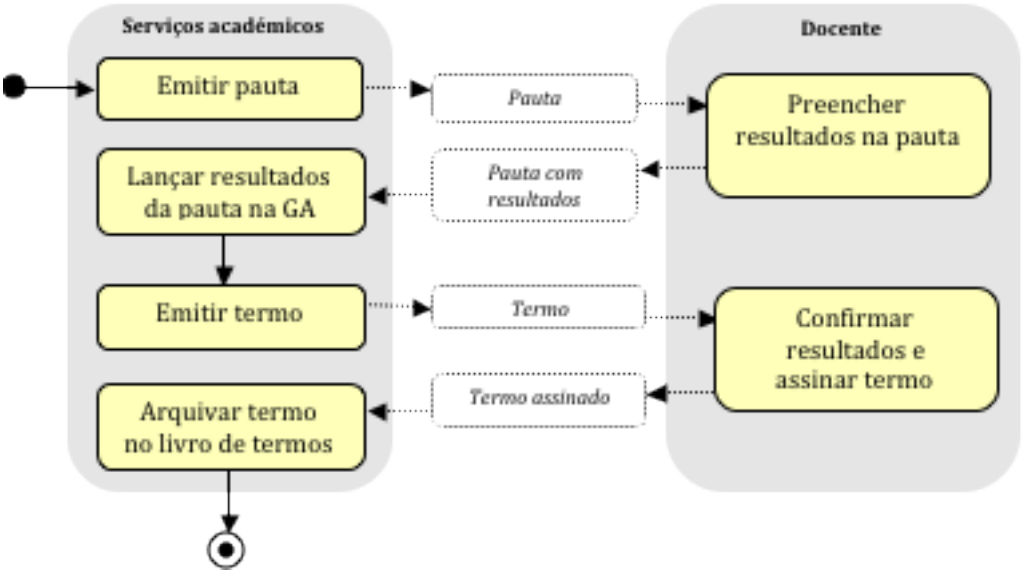


Figure 2.1: Lifecycle of grading record

The process described above, assures that both authentication and authorization are valid in the information system and asserts the user's identity when physically signing the document thus offering non-repudiation from the signer, effectively committing himself to the produced document.

As previously specified, the level of complexity of the information system's workflow as a whole creates the necessity of reducing the set of functionalities as to subdivide our problem by employing a divide-and-conquer strategy. Providing a flexible enough solution to the intricacies of the grading problem and defining a model to the largest

trust-delegation chain in the information system inherently solves some of the less complex workflows not requiring such a long-term validity. Using the same approach provides an anchor-point to a general solution that can be applied in the most general contexts.

2.3.3 Risks in the grading process context

The grading process as described in the previous subsection takes effort on legally binding the document to the issuer, mitigating some of the risks by requiring a presential signature after the document is materialized. The problem with this method arises from the necessity of long-term validity in archived documents.

While document signatures are validated when issuing a document, not only the cross-validation between the archived documents and the data contained in the information system is rarely done mainly because the verification of signatures is a complex time consuming process which requires a level of expertise not in the scope of staff training.

This loss of trust in the archived information over time creates serious risks that have to be addressed:

- Forged documents, if present on the archives, might therefore be considered valid and trusted by the institution services.
- Non-repudiation of old documents by the issuer
- Mismatch of information contained in the archives and in the information system not detected
- Grade certification issued relying on an untrusted source

Mitigating these risks while maintaining the same workflow is an inefficient costly process. Even the simpler cross-validation between archives and the information system would require staff training to spot forgeries, adding complexity to the process and eventually, delays.

Chapter 3

Technical Background

3.1 Public Key Cryptography

Public key cryptography is a cryptographic scheme that relies on a pair of asymmetric keys for the crypto operations. Although the keys are mathematically related, the private key cannot be easily deduced from the public key. Both keys have different purposes, while the public key serves as the encryption key, the private key is used as decryption key.

The public key encryption scheme must retain two properties :

- Completeness

  The encryption function and decryption function are inverses.

  Given any message and a key pair (k1,k2):

  $$\boxed{E_{k1}(D_{k2}(m)) = D_{k2}(E_{k1}(m)) = m}$$

- Semantic Security

  The encryption of different plaintext is computationally indistinguishable.

Since the public keys used for encryption cannot be used to deduce the private key, the public key can be as its name indicates published without affecting the security of the scheme. This provides a solution to the requirement of secret key sharing in symmetric encryption, allowing anyone with access to the public key to effectively encrypt data only decipherable by someone who has access to the private key.

Although it is not necessary to exchange secret keys, the security of the encryption scheme has to rely on the trustworthiness of the obtained public key as pointed out in Diffie and Hellman's seminal paper [23] .

The necessity of trusting the distribution of private keys to different actors is solved by using a Public Key Infrastruture.

3.1.1 Public Key Infrastructure

A public key infrastructure is responsible for certificate storage and management. It provides the ability to issue, revoke, store, retrieve and assert trust on certificates by introducing the concept of Certification Authority. A certification authority, also known as CA, is a trusted entity which has public certificates pre-shared with different software vendors (root certificates) and it retains the ability to sign other certificates using its private key, extending their trust to the signed certificate by creating a chain of trust. Having such an important role in asserting the trust to other certificates, the CA has a central role in a Public Key Infrastructure by leveraging its trustworthiness and reputation. A certification path, also referred as a trust chain, is the list of certificates used to authenticate an entity. The verification process of the trustworthiness of a certificate is determined by verifying the authenticity and trustworthiness of all the previous certificates confiding trust as described in Figure 3.3.

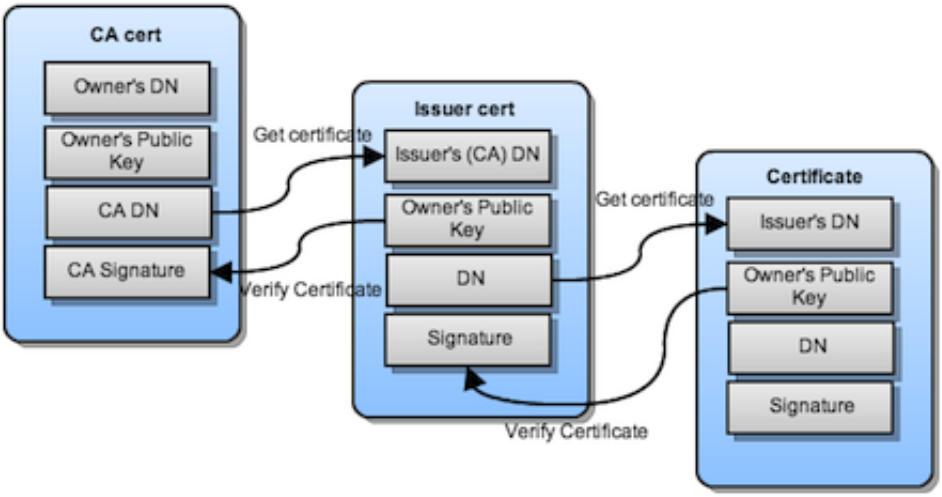


Figure 3.1: Certificate validation in a trust chain

3.1.2 Digital Certificates

A certificate, as defined in the Merriam-Webster dictionary, is a document containing a certified statement especially as to the truth of something. Different properties help verify the identity of the certificate holder (signature, photograph) and embedded security features establish the integrity of the document thus offering a degree of protection against forgeries. Digital certificates are therefore the digital equivalent to their physical counterparts. Actually they serve as containers to bind a public key with an identity or a set of attributes.

The X509v3 certificate commonly used in SSL contains the following attributes:

- *Version* - The version field contains the version of the certificate. The original 1988 version has the value of zero. This field facilitates the implementation of changes in the certificate format over time.

- *Serial Number* - This field contains an unique identifier for each certificate generated by the issuer. The issuer is therefore responsible to guarantee that no colliding certificates (with same serial numbers and Distinguished Name) are generated.

  The serial number is used when identifying revoked certificates in a Certificate Revocation List.

- *Signature* - This field identifies the algorithm used by the Certificate Authority to sign the certificate.

- *Signature Value* - The signature value contains a digital signature computed upon the ASN.1 encoded certificate.

- *Issuer* - The issuer name provides a representation of it's issuers identity in the form of a DN. The issuer name is used to select the appropriate issuer in order to validate a certificate.

- *Validity* - Identifies the duration interval in which a certificate is considered valid by carrying a pair of date and time indications.

- *Subject* - The subject identifies the entity associated with the public key stored in the subject public key field. The field must contain an X.500 distinguished name.

- *Subject Public Key Info* - This field contains the public key and the identification of the algorithm with which the key is used.
- *Issuer unique ID* - This field must only appear on versions above two and should not be used. This identifier permits disambiguating certificates with identical issuer names.
- *Subject unique ID* - This field permits the disambiguation of certificates with identical subject names.
- *Extensions* - This field must only appear in version 3 of a X509 certificate. If present, contains a sequence of one or more certificate extensions. This additional extensions provide the methods for associating additional attributes with users and public keys and for managing a certification hierarchy.

The certificate signature determines that the certificate was vouched for, and it is only able to guarantee that the signed identity information is bound to the specified public key.

Asserting the security of private keys at all times is obviously infeasible, so mechanisms able to revoke the digital certificates are required.

In RFC 5280 [24] two states of revocation are defined:

- *Revoked* - This is a final irreversible state. The certificate is considered no longer trusted and will be present in all future certificate revocation lists
- *Hold* - This is an intermediary state. The certificate is invalidated temporarily and may be removed from further revocation lists if requested.

There are many reasons to revoke a certificate and they can be specified in the revocation entry:

- *unspecified* - unspecified reason for certificate revocation.
- *keyCompromise* - the private key of the certificate was compromised.
- *cACompromise* - the private key of the certificate issuer was compromised.
- *affiliationChanged* - the certificate subject does no longer correspond to organization.

- *superseded* - A new certificate replaces the certificate.
- *cessationOfOperation* - The certificate subject does no longer require the certificate.
- *certificateHold* - the certificate is revoked temporarily.
- *privilegeWithdrawn* - privileges granted to the certificate's subject have been withdrawn.
- *aACompromise* - it is known or suspected that aspects of the Attribute Authority validated in the certificate have been compromised.

When validating a certificate, verifying its revocation status can be achieved using different methods: using a certificate revocation list containing a list of certificate serial numbers parsed by the verifier, or using the Online Certificate Status Protocol. This protocol uses ASN.1 encoded messages communicated over the HTTP protocol to return the status of the certificate.

3.1.3 Digital Signatures

A Digital Signature is a mathematical scheme based on asymmetric cryptography used to demonstrate the authenticity of a digital document, by relying on a private key for generating a signature of the document which can be verified by the public key. Establishing the document's authenticity implies the properties of non-repudiation ,authentication and integrity are satisfied.

- *Authentication* - Authenticating the sender of a message or document with a high degree of confidence is important in some occasions. In digital signatures, when a given key is bound to a user, the authenticity of origin can be established with a digital signature.
- *Integrity* - To guarantee that each message or document is not altered in transit and assure the accuracy of the information. Digital signatures intend to provide this degree of confidence on data integrity by using cryptographic hashes, relying on the computationally infeasibility of generating data with the same hash values.
- *Non-Repudiation* - This property assures that an entity that signed a given message is bound to it and cannot in a later stage deny having signed it.

3.2 Smart-card

A smart card is a device that contains an embedded integrated circuit chip. Although they are usually associated with their most common types, the smart card with embedded secure micro-controllers, there are different kinds of smart cards available, providing simple I/O functions or supporting on-card functions such as encryption and mutual authentication and interacting with the reader using specific protocols. When dealing with micro-controller smartcards, the ability to generate and store certificates in a tamper-proof environment with irrecoverable keys inside the secure element and perform on-card functions provides an effective control of the keys used in cryptographic operations or other cryptographic objects being stored in the smartcard filesystem. This ability to ensure that keys cannot leave the card, provides an effective means to ensure non-repudiation.

3.2.1 Portuguese National Identity Card

The Portuguese National Identity Card [25] is an EAL4+ compliant smart card retaining the ability to produce qualified signatures. The smartcard contains a micro controller supporting the latest Java Card version, and a 64K EEPROM supporting on-card cryptographic functions:

- Signature and Verification of 1024 bit RSA
- DES and 3DES
- MD5
- SHA1 and SHA256
- Message Authentication Code (MAC)

The certificates contained in the card (Authentication and Signature certificates) are issued by the respective sub-certificate authority. The sub-certificate authorities are branches of the Common Certificate Authority for the Portuguese government.

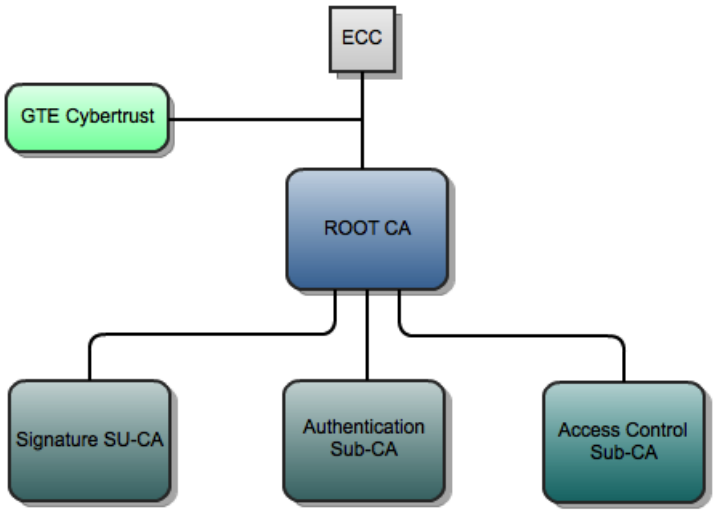


Figure 3.2: Portuguese Government certification chain

In respect to security, the smart card is known to be resistant to different known attacks on smartcards like Hardware Attacks, Timing Attacks, Simple Power Analysis and Differential Power Analysis [25].

3.2.2 University of Porto Smartcard

The University of Porto identity card is an Optelio Contactless D32 R5, PayPass certified smartcard produced by Gemalto [26]. Complying with the Javacard 2.2.1, the smartcard contains a Cryptographic co-processor and a 32K EEPROM providing a dual-interface both contact and contact-less.

The contact interface complies with ISO7816 standard, supporting both character and block level transmission protocols (T=0 and T=1).

The contact-less interface complies with ISO 14443 -2,-3 and -4 standards, operates at a frequency of 13,56 MHz and includes a Mifare 4K emulation.

The Optelio smartcard supports RSA (up to 2048 bits), DES/3DES and SHA-1 cryptographic algorithms and contains different installed applets, namely:PayPass M/Chip4 (Mastercard)[27], VSDC2.7.1 (Visa), DualVSDC (Visa), Classic IAS V3 (GemSAFE) (PKCS#11 PKI application), WG10 and Welcome Realtime (WRT) XLS V7 (Xena).

3.2.3 University of Porto Certificates

The University of Porto smartcards come uninitialized and do not contain previous certificates, thus relying on Terena as a Certificate Authority to issue signed certificates for advanced digital signatures. To support self-service card-initialization,an Applet which matches the user to its Portuguese Identity card using the card authentication certificates provides the users the ability to request a signed certificate for their role inside the institution. By requesting the users to authenticate themselves with a government issued ID and assuming these are registered users at the university, the applet can bind both identities (citizen's and user) and allow a new certificate to be requested and immediately imported into the smartcard.

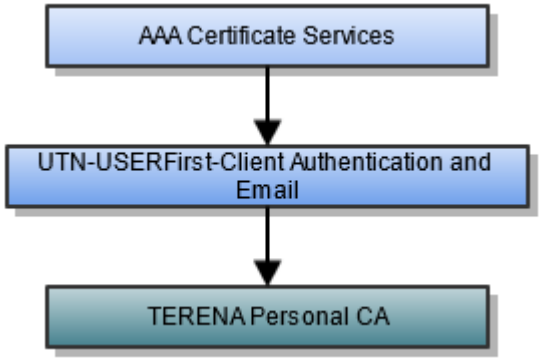


Figure 3.3: Terena Personal certificates chain

3.3 XML

The Extensible Markup Language or XML is a markup language defining a set of rules to encode documents in both human and machine friendly format. Derived from SGML and defined in the XML 1.0 specification by the W3C, the XML provides a very flexible text format. Using the XML document's flexibility we can replace the entire relational database scheme with a XML-centric database containing signed XML documents as described in section 3.4.

3.3.1 XML signatures

XML Signatures are digital signatures defined for the XML document format in the xmldsig namespace providing integrity assurance, message authentication and signer

authentication for any kind of data objects.

The message integrity is guaranteed by the computation of a a message digest using a cryptographic hash algorithm and the difficulty of finding two meaningful collisions of messages, ensuring that an attacker could not change the message without breaking the digest. Although the digest reflects message tampering, a message could be modified in transit and a corresponding hash computed for the modified message.

In order to guarantee that the digest cannot be modified after the message is signed, the digest value is encrypted with the private key of the sender. Using the public key available from the sender public certificate, the recipient of the message can decrypt the digest and verify the message.

The XML signatures can be present in the same document as enveloped, enveloping or outside the document (detached signature).

- *Enveloping* - The signature wraps the signed elements. In this method the signed elements appear inside an Object tag inside the signature element and are identified using a Reference.

- *Enveloped* - The signature is produced inside the element that contains the data to be signed. Producing this signature types requires a special care not to include the signature fields when calculating the signature value.

- *Detached* - Detached signatures, as the name implies, sign elements that are external to the signature element. This signatures can also be present in the same document but having both the signature element and signed element as siblings.

3.3.2 XML Canonicalization

Digest algorithms are agnostic to the XML syntax and work over a sequence of bytes. This unawareness of the syntax implies that two syntactically equivalent messages will produce two different signatures.

Line endings, for instance, are represented in a platform-dependent way, producing different XML representation in different architectures and obviously, different message digests.

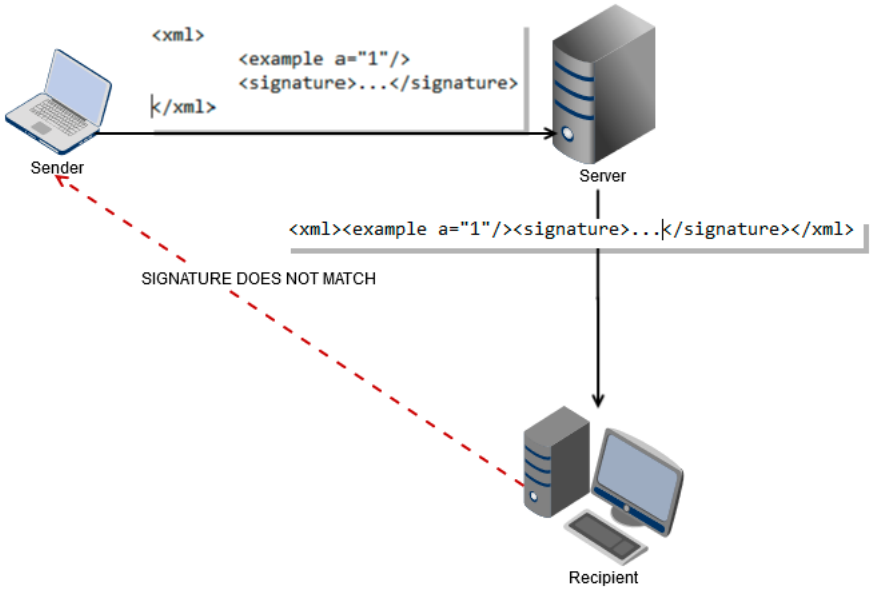


Figure 3.4: Linefeed signature mismatch

In order to address the problem of serializing XML documents and inadvertently producing different documents, a Canonicalization Specification proposes a transformation denominated canonicalization that standardizes the document to a single serialized representation. The canonicalization takes a well-formed document and produces a syntactically equivalent XML document and further canonicalizations would always produce the same document. To produce a canonical form, a set of rewrite rules must be defined which, when applied in any order the result is always a canonical form,Table 3.1 presents such a set of rewrite rules.

Table 3.1: Canonicalization scheme for a XML document

Possible Variations	Canonicalization
Character Encoding	Convert to UTF-8 if any other encoding is used.
Character sequence used for line breaks	Standardize to #xA (line feed).
Optional XML declaration	Remove.
Optional DTD declaration	Remove.
Character references	Expand character references.
Use of CDATA sections rather than escaping special characters	Replace all CDATA sections with the equivalent character content. Special characters are replaced with character references.
Use of empty element tags	Replace all empty tags with a start/end tag pair. `<a/> replaced with <a> </a>`
Whitespace within tags	Remove. `< a > replaced with <a>`
Whitespace before the root element and after the end of the document	Remove.
Whitespace in element content	Retain.
Choice of quotations marks in attribute values	Always use double quotes to delimit attribute values. Single quotes (') replaced with double quotes (").
Attribute values	Use attribute value normalization defined in XML specification.
Default attributes	Add all missing attributes for which default values are available.
Attribute order	Sort all attributes by their namespace URI and local names in lexicographically ascending order. Attributes with no namespace come first. Namespace nodes come before other attributes, and the default namespace node comes first.
Namespace declaration	Normalization depending on either inclusive or exclusive canonicalization specification.

The rules defined in Table 3.1 can be applied in any order producing a document in canonical form. Although most of the rules are easily applied, the namespace declaration canonicalization introduces two different possibilities for rewriting namespaces, inclusive or exclusive canonicalization.

3.3.2.1 Inclusive and Exclusive canonicalization

Extracting a document subset and inserting it into a different context may cause problems with signatures because canonical XML includes the document's ancestor namespace declarations and attributes within the XML namespace. Take for instance a document in canonical form:

```
<set>
<element>example</element>
</set>
```

The document subset can be encapsulated in an envelope.

```
<envelope xmlns="http://www.example.pt" xml:lang="pt">
<set>
<element>example</element>
</set>
</envelope>
```

When inclusive canonical form is obtained from the enveloped document, the document is a little bit different:

```
<set xmlns="http://www.example.pt" xml:lang="pt">
<element>example</element>
</set>
```

The original document does not have the same canonical form even though it is the same document thus having a different digest.

While inclusive canonicalization includes the the context of the subset's ancestors , and in order to address the previously identified problems, exclusive canonicalization was devised.

In exclusive canonicalization, namespace declarations and the attributes in XML namespace of a a subset ancestor are excluded from the canonicalization process.

Taking the previously described example, the canonical form of the enveloped set would be :

```
<set>
<element>example</element>
</set>
```

As we can observe, the document's exclusive canonicalization produces a canonical form identical to the initial and intended form.

3.3.3 XMLdsig structure

An XML Signature element structure can be described with the following elements:

- SignedInfo

  The SignedInfo Element is the root of the structure used to identify the methods used in the verification of a given signature.

  - CanonicalizationMethod

    This method provides information for the transformation of the information into a canonical form to be used in the signature operations.

  - SignatureMethod

    This element identifies the algorithm to convert the canonical form of the element into the signature value. This method is a combination of an Hashing algorithm and a signature algorithm such as RSA-SHA1.

  - Reference

    Each of the multiple reference elements that may be present includes information on the digest methods and corresponding result values over the referenced data object.

- Transforms

  The Transforms element is optional, and may contain a list of transform elements, passing the results through the different transform elements which contain

the attribute defining the algorithm and respective content parameters operating on the node set.

- DigestMethod

  This element identifies the digest algorithm to be applied to the object being signed.

- DigestValue

  Containing the encoded value for the digest, the DigestValue element content is encoded using Base64.

- SignatureValue

  As the name implies, this element contains the final signature value encoded in Base64.

- KeyInfo

  This optional element enables the recipient of the document to obtain the key required to validate the signature. This information may be composed of keys, names, certificates or other public key management information required to obtain the needed key. Since the element is optional and when this element is not present, it is assumed that the party validating the document is able to identify the needed key in the application context.

- Object

  This element is typically used in enveloping signatures where the signed element must be contained inside the signature element. The object's id is referred from a reference element either in the SignedInfo element or Manifest.

```
<Signature>
  <SignedInfo>
    <CanonicalizationMethod />
    <SignatureMethod />
    <Reference>
       <Transforms>
       <DigestMethod>
       <DigestValue>
    </Reference>
    <Reference />
  </SignedInfo>
  <SignatureValue />
  <KeyInfo />
  <Object />
</Signature>
```

Code Block 3.1: "xmlDsig structure"

An XML document may contain multiple signatures implying that multiple signature tags may be present in the document. These signatures can be produced at different times by different entities who may sign different elements. The widespread usage of signed XML documents identified several problems with this signature specification, namely the non-repudiation and long-term validity of signed documents. To address these issues with the XML-dsig specification a new specification (XadES) was developed by the European Telecommunications Standards Institute extending the previous XMLDSIG specification.

3.3.4 XaDeS

The XML Advanced Electronics Signature specification defines different profiles that can be used to sign XML documents extending the existing XMLDsig by providing qualifying information and non-repudiation to the signature.

These profiles can be defined in levels. Each level not only inherits the requirements of the parent but requires more information effectively adding stronger assurances to the signed data.

The simpler form of XadeS profile adds the following elements to the previous specification:

- *QualifyingProperties* - Element containing the qualified properties, a sequence

containing one or more elements of type SignedProperties and Unsigned Properties.

- *SignedProperties* - Element containing the signed qualified properties.
- *SignedSignatureProperties* - Element containing the properties that qualify the XML signature specified with the Target attribute of QualifyingProperties.
- *SigningTime* - The SigningTime property specifies the time at which the signer performed the signing process.
- *SigningCertificate* - This property contains references to certificates and digest values computed on them, preventing the substitution of the certificate.
- *SignaturePolicyIdentifier* - The signature policy identifier is a signed property qualifying thesignature.
- *SignatureProductionPlace* - Element that may contain the place where the signer was at the time of signature creation.
- *SignerRole* - Signed property qualifying the signer, specifically the role in which the signature was created.
- *SignedDataObjectProperties* -Ccollection of signed XML elements with properties individually qualifying signed data objects.
- *DataObjectFormat* - element providing information that describes the format of the signed data object.
- *CommitmentTypeIndication* - Signed property qualifying the signed data objects, specifying the commitment type.
- *AllDataObjectsTimeStamp* - Time-stamped ds:reference elements within SignedInfo referencing whatever the signer wants to sign except the SignedProperties element.
- *IndividualDataObjectsTimeStamp* - Element containing the time-stamp computed before the signature production, over a sequence of some ds:Reference elements within the SignedInfo element.
- *UnsignedProperties* - Element containing a number of properties not signed by the XMLDsig signature.

- *UnsignedSignatureProperties* - element that may contain properties that qualify XML signature itself or the signer.

- *CounterSignature* - Element containing a counter-signature.

The described Xades profile provides integrity protection and basic authentication.

In order to offer stronger non-repudiation, five more profiles are defined by the specification.

- Timestamped (Xades-T)

  This XML Advanced Electronic Signature profile includes a timestamp to ensure non-repudiation by introducing a new element - SignatureTimeStamp. Producing a document with a timestamp requires therefore a TimeStamp Authority (TSA) , binding the signature of the document to the date obtained from that external entity.

- Complete (Xades-C)

  This profile adds references to the required data supporting signature validation such as certification path and revocation information.

- Extended (Xades-X)

  The extended profile requires a timestamp either on the whole signature (as in Xades-T) or in the information supporting signature validation (Xades-C). By providing this possibility, the verifier can be assured that the information about the certification path and revocation status was valid at the specific time the timestamp was added to the document.

  Assuming that a Certificate Authority can be compromised and its certificates revoked, the timestamp established the validity of the signature by assuring that the timestamp is earlier than the revocation event.

- Extended Long-Term (Xades-X-L)

  Information required to validate documents may become obsolete on the long-term. To solve this problem, this profile provides the possibility to add the certification path and revocation data to the document ensuring that document validity can be asserted in the long-term.

- Archival (Xades-A)

  This profile adds a timestamp to the information used by the Extended Long-Term profile.

  As computational power increases, keys and other cryptographic data may become weak. The ability to ascertain the integrity in the archives requires, therefore, the ability to produce stronger signature over time. Xades-A considers this problem and by providing the support to generate multiple stronger timestamps guarantees that the integrity of the information to validate the signatures is not compromised.

3.4 Marklogic Database

Marklogic[28] is a document-centric, transactional,search-centric, structure-aware and schema-agnostic database server. Using XML document as its core data model and providing XQuery and XSLT support, marklogic is a perfect tool for our architecture. Although it is possible to retain the OracleSQL database, relational databases have table-centric models having higher complexity with an obvious impact in performance.

Marklogic supports the full set of ACID properties:

- *Atomicity* - sets of changes only happen as a whole.
- *Consistency* - system rules are enforced so there are no colliding identifiers between documents.
- *Isolation* - uncompleted transactions are not otherwise visible.
- *Durability* - once a transaction has been committed it will not be lost even in the event of a power loss.

Indexing both text and structure, the two can be queried together efficiently facilitating search of highly structured documents.

3.5 Signserver

SignServer is an application framework performing cryptographic operations for other applications. [8]. Based on loadable modules that perform a wide range of operations,

this framework is able to perform signatures on different types of file formats,namely PDF,XML,ODF,OOXML,MRTD,CMS and supports online document validation.

The modules can be used using HTTP or webservices interfaces.

The framework defines three different processable services:

- *Signers* - responsible for signing or otherwise processing the requested data.
- *Validation services* - services enabling the validation of a certificate against a set of backed issuers.
- *Group key service* - framework enabling management and distribution of group keys.

In order to perform sets of operations at defined intervals of time, the framework provides plug-ins denominated Timed Services. PrimeKey's sign server already supports different kinds of hardware security modules and PKCS#11 devices, thus providing the required flexibility to implement a HSM-backed signing module as denoted in Figure 3.5.

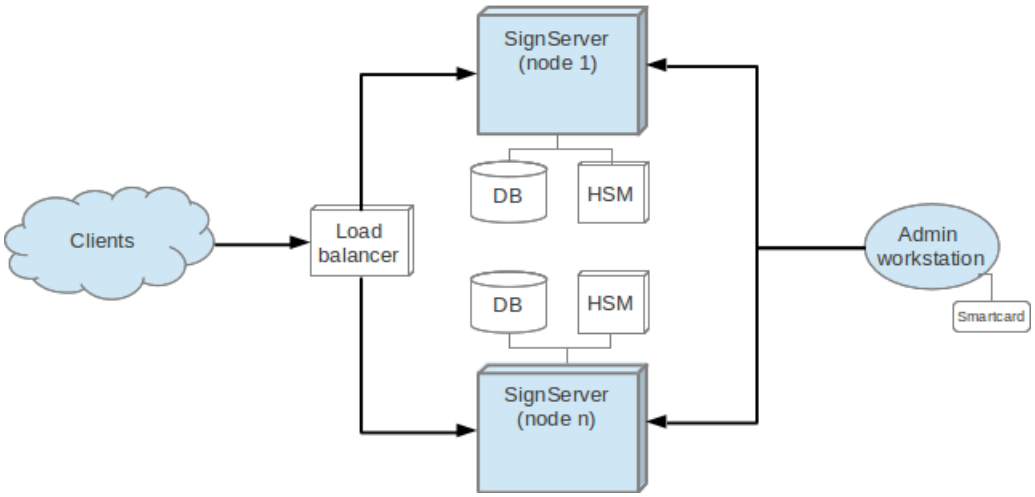


Figure 3.5: Signserver Framework architecture

While Primekey's Signserver supports XML it did not support XaDeS with the XML signature module supporting only xmlDsig. Since signserver is an opensource project, the development of a new module providing the required XaDeS support not only is possible but the modules source contained in the project can serve as the basis for development.

The fact that signserver is an opensource project made it the inevitable choice for our architecture, providing the ability to create complex modules aware of the in-house trust-delegation chains which establish the trust anchors for any document.

3.6 HSM

A Hardware Security Module is a device containing one or more secure cryptoprocessors, providing strong authentication when accessing keys, secure key storage by using tamper-resistant secure elements and accelerating cryptographic processes by using dedicated hardware to offload the cryptographic operations from application servers.

The private keys are generated on-board and remain irrecoverable inside the secure elements, providing both logical and physical protection to the cryptographic keys. Tamper protection is, therefore, a key element in Hardware Security Modules which support a multi-part user authorization schema enforcing cooperation when dealing with sensitive materials like private keys.

Physically protecting the keys in hardware requires a mix of techniques:

- *Tamper Resistance* - Packaging the hardware in a hardened shell casing, making access to the inner parts difficult.
- *Tamper Evidence* - Protecting the device with seals and labels designed to visually identify any attempt to open the device.
- *Tamper Detection*

  Using a conductive mesh embedded with the packaging provides the internal circuits with a method to monitor different properties that would signal tampering.
- *Tamper Response* - Using an active response when tampering is detected, either by logical destruction of data or, in military condition,s physical destruction of both the equipment and data.

The HSM acquired by the University of Porto is Safenet's Luna SA 5.0 , supporting a PKCS#11 API and a wide range of cryptographic algorithms:

- *Asymmetric:* Diffie Hellman (1024-4096 bit) , RSA(512-8192 bit) , DSA (512-1024 bit),ECDSA signing with over 50 standard curves.

- *Symmetric:* 3DES,AES,RC2,RC4,RC5,CAST
- *Hashing:* MD2,MD5,SHA-1,SHA-2 (150,224,256,512 bit)
- *Message Authentication Codes:* HMAC-SHA1 and HMAC-SHA256

In order to enforce division of operational roles, the HSM requires the use of PEM keys in order to perform some management and maintenance operations:

- *Security Officer* - configure HSM and change device's security policies.
- *User/Partition owner* - activate partitions and generate keys.
- *Domain* - control and define the security domain, replicate and backup keys.
- *Remote PED Auth* - assure logical trusted path to remote PED
- *Tamper Recovery* - recover from a tampered status, enable safe transport.

This HSM also provides mechanisms to avoid unilateral action by key holders by allowing the requirement of multiple PEM keys to perform operations.

The application server is connected with the HSM using Network Trust Link using four layers of security: IP Address, TLS with client authentication, Administrator controled trust database and partition access password.

Figure 3.6 describes the overall architecture.

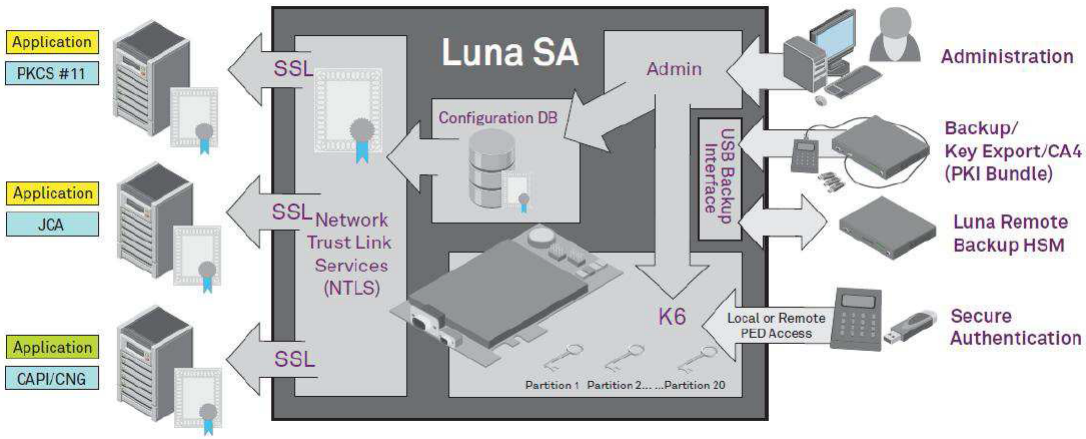


Figure 3.6: LUNA SA solution architecture

Chapter 4

Comparison with similar systems

Information systems have for long strived with the importance of document dematerialization as a means for cost-reduction.

While documents have been moving to digital archives, processes and document work-flows had problems retaining the same level of trustworthiness as their physical counterparts. To address this concern, multiple solutions have been proposed from which we will explore in detail the University of Murcia e-government solution[29].

The coincidental similarity of the University of Murcia and our solution, provides a good starting point to evaluate the differences and impact of both solutions in order to achieve the main goals.

4.1 University of Murcia e-Government

The proposed solution by the University of Murcia intends to provide transaction security and document management through a secure platform, by delivering two groups of services.

4.1.1 Security Services

- *Signature Service* - This service provides the ability to sign and verify documents.
- *Timestamp Service* - In order to ascertain proof that a document existed at a certain time, a timestamp service is provided.

- Certificate Validation Service

  The validation of certificates used in digital signatures is performed using this service.

- *Accounting Service* - The accounting service is responsible for logging activities of the previous services.

4.1.2 Documentary Management

The document management service set represents the group of services required for secure generation, archive and query of documents while preserving electronic evidences.

- *Archiving Service* - Stores electronic documents in the university documentary database while assigning an unique identifier.

- *Query Service* - Locate and Retrieve documents using the unique identifier.

- *Document Generation Service* - A service for creating automatically documents.

4.1.3 Client Architectures

In order to generate signatures and perform authentication using the University of Murcia Smartcard, client-side applications have been developed by the University of Murcia. With cross-platform operation as a goal, the architecture supports both the Windows Crypto API and CAPICOM within the Microsoft Windows environment and PKCS#11 with NSS when using a Linux Operating System as depicted in Figure 4.1.

The UMU crypto-Applet verifies the client's browser and operative system and provide digital signature support using the MSCAPI Java provider or the NSS Provider accordingly and transparently to the user.

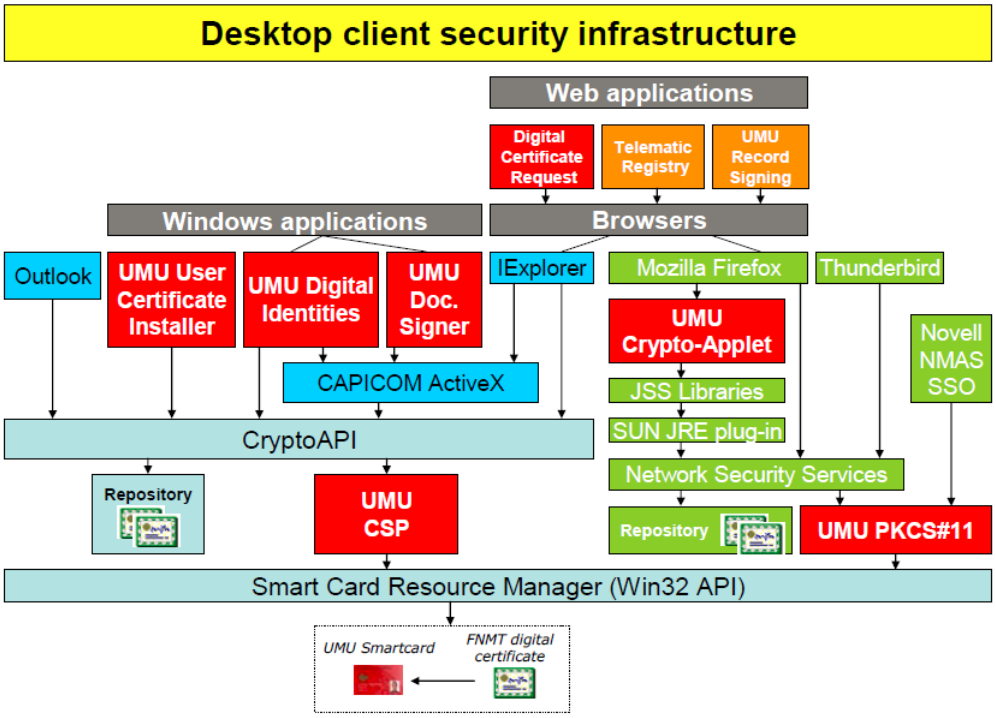


Figure 4.1: Client Architecture

Figure 4.1 displays the separation that occurs when using different browsers, this approach requires the smartcard middleware to be previously installed and the use of non standard methods to access the cryptographic interface. The CAPICOM [30] interface is discontinued although as yet supported by most of Microsoft Operating Systems.

4.1.4 Electronic Registry

The electronic registry provides the ability to send electronic documents and forms to the University retaining the same level of trustworthiness as the physical registry. This application intends to retain both authenticity, integrity and non-repudiation properties of the transactions.

An example of such a transaction is shown in Figure 4.2

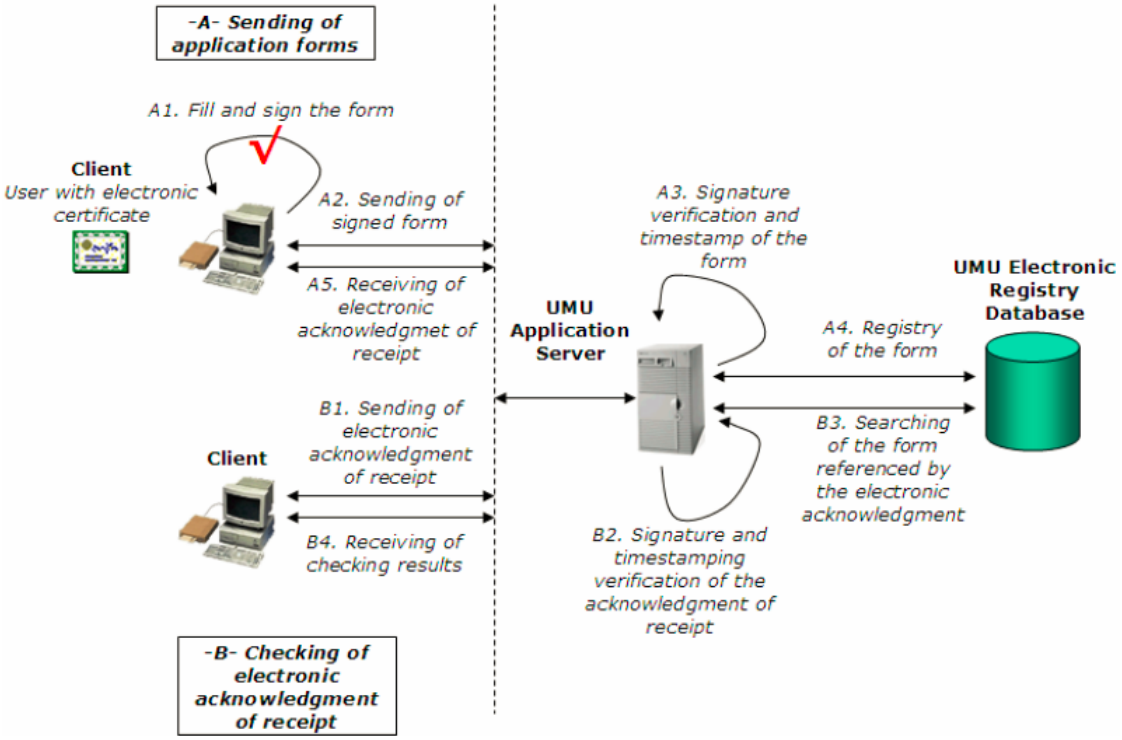


Figure 4.2: Sending an application form

The client fills and sign a form and sends to the application server. The application server after validation and timestamping, records the form in the registry database and delivers an acknowledgement of receipt. A previously sent form can be verified by sending the acknowledgement receipt to the application server which verifies the timestamp and signature, searches whether the form is present in the registry database and returns the result to the client.

4.1.5 Electronic Marks Certificate

The electronic marks certificate service extends on the Electronic Registry to provide professors the ability to perform marks certificates without having to deliver documents containing an handwritten signature. The electronic marks are created as an XML document, signed by the professor, providing the ability to verify the authenticity and integrity of each mark through the digital process.

4.1.6 Analysis of solution

The solution implemented at the university of Murcia although providing a good starting point to develop an environment for securing digital documents presents some problems that must be addressed.

- *Guaranteeing the long-term verifiability of the document author authorization to produce the document* - The document is produced with a context that may be considered volatile, and cannot be easily audited in the future.

  This problem arises when a user produces a colluding document which the original author is unaware off, the document would be signed, timestamped and archived. A future audit, if it does not possess a reliable information source containing the authorized identity to produce the document, cannot disambiguate which document is to be considered valid, since both authors had valid certificates at the signing time.

- *Non-repudation only for the author of a document* - The effort to produce a document receipt is not enough to provide mutual non-repudiation. A process of dispute arbitration in which one of the challengers has set itself as the trusted party is flawed by design. Mitigating this problem requires the presence of a trusted third-party, with mutually agreed jurisdiction, producing signatures and arbitrating disputes [31].

  The receipt of acknowledgement, cannot provide mutual non-repudiation unless a trusted third-party is relied upon.

Chapter 5

Proposal

The development of a system that can handle the identified problems while retaining strong confidence of document integrity requires a careful workflow design and architectural changes to the existing information system.

In Chapter 2 we introduced multiple risks that could lead to the untrustworthiness of the system data. In order to minimize these risks a solution is described in detail with sections identifying which problems are being tackled.

5.1 Document-Oriented Data

The information contained in the information system is document-oriented. Data cannot be viewed as single fragmented piece of information mainly because it is mostly produced in-context, structure-dependent and bound to other information blobs.

An example of such document relates to student grading,where each grade is bound to a specific student, the document's date, course, class and professor. Given that a student can take the same class in different time periods and be evaluated more than once by different professors, committing to such information's trustworthiness implies committing to the entire structure and not simply to the grade alone. Furthermore, a grade can be attributed in multiple ways, such as an equivalence in a student mobility program. Identifying the origin and legitimacy of such a document, requires not only signed atomic data, but also information of the context in which it was produced.

Retaining such a document structure is not convenient when dealing with relational

databases that imply the use of their formal data structures.

While the current information system materializes documents in order to retain this structure information and trust, this physical support is expensive and workflow ineffective as workflow is concerned. To address this problem, a shift to XML documents is required, that by their semi-structured nature provide us the ability to retain the complex structures used during document production and by leveraging digital signatures ascertain integrity and non-repudiation.

5.2 Data persistence

While the current information system stores its information in a relational database, in the previous section problems were identified when using relational databases and a solution that fits the specificity of the information system's environment was provided. The move to XML documents introduces a new requirement: XML document storage.

In order to address this requirement, a change in the overall architecture of the information system was introduced, an XML database was added providing the information system with the ability to store and operate over XML documents.

5.3 Document Trust

In subsection 2.3.3 we identified the threats leading to the loss of trust in document's integrity in the current system workflow. The move to XML documents requires, therefore, that we can assure the integrity of every document stored in the database and mitigate the risks that were also present in the previous scheme.

To trust the documents contained in the database we must retain the ability to:

- Authenticate the person who produced the document
- Provide non-repudiation of document's authorship
- Assert that the document was not tampered with
- Assure that the document was produced in a given timeframe

To address these requirements, the documents must be digitally signed by a certificate, bound to the user (authentication, non-repudiation and integrity assurance) and timestamped by an external entity to provide a trustable date source. While simple documents can be produced in such a way, in an academic environment with shifting hierarchies we must provide a method to assure that the given user was indeed acting on behalf of the university when producing the given document. This delegation of trust in the individual to produce a document requires another digital signature from the institution itself assuring that at the time of document production a user was in fact able to create such document.

Figure 5.1 describes the required properties in such a document:

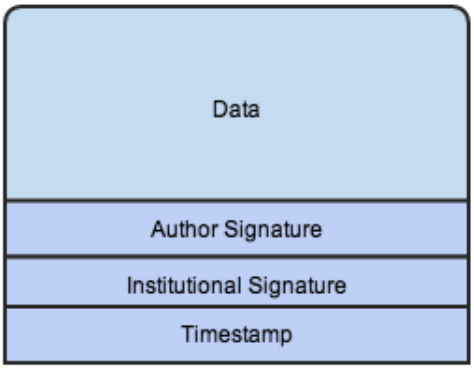


Figure 5.1: Document Structure

Delegating trust is itself an anchor to the entire trust of the document and when dealing with complex hierarchies entrusting a user to perform a given activity requires a long chain of trust delegation. To cope with this problem, we must introduce a document that can describe such trust chain.

5.4 Hierarchies and trust delegation

Trust delegation is implicit in the current information system as users are authenticated by their credentials and attributed a set of roles that provide them with the ability to perform a set of functions. Producing documents and assuring that the producer had the legitimacy to perform it extends the trust past the document and requires a trusted document describing the role assumed by the user. In order to simplify the development process and minimize the impact of changing roles, the current roles present in the system were assumed and defined in an XML document describing

the user and his role. Documents describing this chained trust delegation are bound together. This delegation chain can be observed in the following picture.

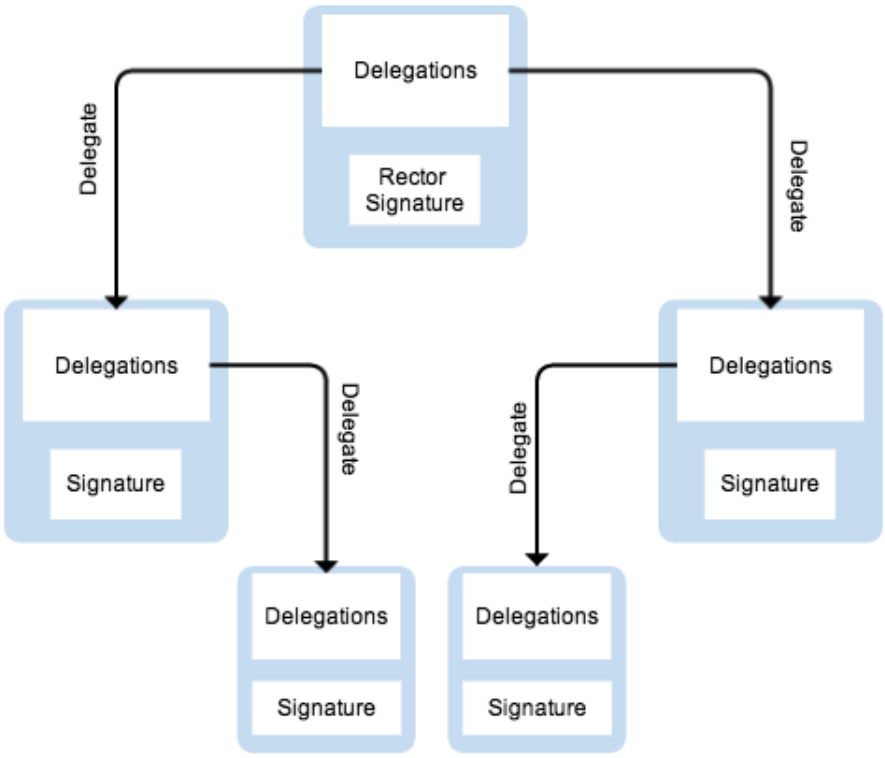


Figure 5.2: Trust delegation diagram

Validating a user's role requires the recursive validation of delegations from the root trust anchor to the delegation that entrusts the user with a given role. Since each delegation document is itself signed, the integrity of the document can be guaranteed. While this diagram of trust delegation is simplified it does not account for temporal delegation, the documents produced contain the date the delegation takes effect and its expiration date. A simplified delegation document describing this temporal delegation can be exemplified the xml document in Code Block 5.1.

Roles are defined by complex types.

Delegated roles are context-dependent. In fact, not only a user can be delegated multiple roles but those roles can be context-specific. A professor assumes his delegated role for a small subset of classes and attains this role by delegation from the director. A user may assume, therefore, multiple roles which demands for example that a director for instance is required to assign himself a set of roles by delegation in case he intends to lecture a class.

```
<delegation>
        <delegate>
                <user></user>
                <role></role>
                <starts></starts>
                <expires></expires>
        </delegate>

/* Signature nodes */

</delegation>
```

Code Block 5.1: "Delegation XML document"

The establishment of a role is generically described as traversing recursively the institutional hierarchy. One example is presented in the diagram Figure 5.3

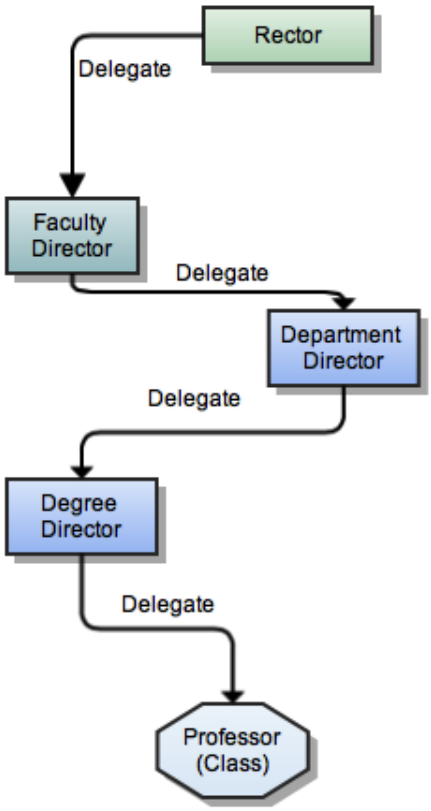


Figure 5.3: Delegation hierarchy

5.5 Grading Documents

Dealing with trust sensitive documents and establishing document integrity through all the document phases (creation, approval, distribution and archival) is a key aspect to ensure the trustability of student records. In section 5.3 we identified the requirements to produce a trusted document and the same principle applies to grading documents. The grading document is therefore an XML document using the XML Advanced Electronics Signature specification, namely XaDes-A as to provide support to long-term archives.

An XML schema is included in the Apendix section describing a structure tying students and the obtained grades and can be modified to different needs.

5.5.1 Professor Signatures

Asserting the authorship of a grading document is very important when assuming that any system can be broken into or subverted in some way. Digital signatures not only provide us with the required proof of authorship but also with the non-repudiation property the document's author cannot refute that he did not produce such a document. Since professors are just temporarily assigned to a class, the author cannot be able, at a later time (when the delegation has expired) to create documents referring to the previous time-frame. To provide a stronger guarantee that the document was in fact produced in the given timeframe, a timestamp is required along with the document's signature.

5.5.2 Institutional Signature

Documents signed only by the professor grading the student may be trustable enough for the institution, but trust must be bidirectional. A professor may require that not only he is bound to the document but also the institutions commits itself to the receipt of the document. Institutional signatures provide the author of the document with a guarantee of non-repudiation and acceptance from the institution. A institutional signature commits the institution not only to the reception of the document but also to the ability of the author to sign the document by assuring that the institution did traverse the delegation chain and accepted the document, thus protecting the author.

5.5.3 External Timestamp

A problem arises when observing that the university controls the entire infrastructure such as the certificates for accepting documents, timestamp servers and so on. In order to attain a stronger trust on the entire scheme and independence of a possible, although improbable, full infrastructure compromise, a timestamp based on an external entity can be used when signing the document with the institutional signature. Anchoring the trust to an external entity which is independent of the institution's public key infrastructure and provides a timestamp for the document assures that the institution cannot later subvert previous documents.

Chapter 6

Implementation

6.1 Architecture

The implementation of such a cryptographic scheme to retain document trustworthiness requires deep architectural changes to the existing information system. Producing client-side signatures requires a Java applet to be delivered to the user so he can perform cryptographic operations on the documents also, a document repository has to be included due to the insufficiency of the current database to store xml documents and a signserver to perform server-side signatures.

In public key cryptography, the entire scheme relies on the security of the private key. To address this concern, a Hardware Security Module is connected to the signserver,thus protecting the private keys and performing cryptographic operations.

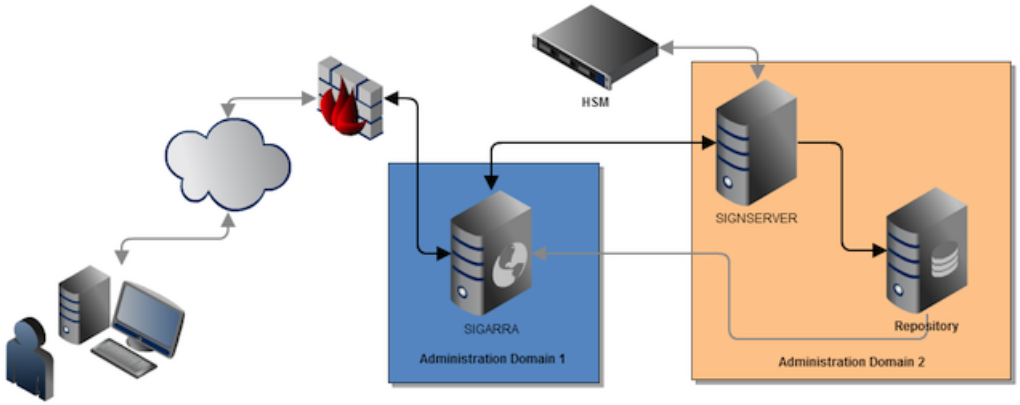


Figure 6.1: New Information system architecture

In the architecture described in Figure 6.1, a separation of administrative domains can be perceived. The information system (SIGARRA) has only direct read privileges from the repository, although transitional write privileges are available by delivering documents to be validated and institutionally signed, which are recorded into the repository. This model provides a strong guarantee that documents in the repository are approved and signed by the institution (offering non-repudiation to the author) and ensures that documents cannot be removed by anyone with administrative rights to the information system.

6.2 User Signatures

Signing a document with the university supplied smartcard requires client-side code capable of using the smartcard's private key. In order to provide support to client-side signatures to XML documents, a Java applet with the ability to digitally sign documents complying with the XML Advanced Electronics Signature specification was developed.

6.2.1 Java Applet

In a complex system, it is not feasible to produce different applets for different contexts. Taking into account that a single applet able to cope with all the contexts can be developed, a more generic approach is devised, where different parameters related to the workflow can be defined.

The applet takes these different parameters:

- Document

  This parameter should contain the document that was generated by the information system to be signed and is encoded in base64. The main reasoning behind this encoding requirement is because XML documents may contain tags similar to the ones used in the HTML page and, in order to avoid problems when rendering such documents, encoding the document in base 64 is a reasonably simple method for solving such cases.

- Signee

This parameter contains the identifier to be matched against the certificates unique name. While the information system may reject documents signed by someone who didn't have privileges to sign a document, users may, by mistake, introduce a smartcard with a certificate not pertaining to themselves. Using this parameter, the applet is able to match the expected signee with the certificate ensuring that the user is indeed using the required smartcard.

- PostTO

  The PostTO parameter contains the service that will be used to consume the signed document. Different document types can be signed in different contexts which have to obviously be signed by different web services.

  The ability to provide a URL indicating where the applet should deliver the signed documents is an approach that provides a generic method to reuse the same applet in different contexts.

In a typical usage scenario, the web server generates a document containing an applet and the parameters above.

The initialization of the applet takes into account the parameters contained in the generated HTML and initializes the internal variables.

```
@Override
    public void init() {
            xmlDOC = this.getParameter("xmlDoc");
            postTO= this.getParameter("postTO");
            signeeUN = this.getParameter("SigneeUN");
}
```

Code Block 6.1: Applet Initialization

The document base64 encoded and passed as a parameter to the applet, which requires the XML to be decoded from the parameter..

```
/*
(...)
        xmlDOC = this.getParameter("xmlDoc");
(...)

*/
public Document decodeB64Document()  throws SAXException,
                ParserConfigurationException, IOException{

        Document doc = null;
        byte decoded[] = Base64.decode(xmlDOC);
        InputStream in = new ByteArrayInputStream(decoded);
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        factory.setNamespaceAware(true);
        DocumentBuilder builder = factory.newDocumentBuilder();

        doc = builder.parse(in);

        return doc;
}
```

Code Block 6.2: Decoding XML document

Signing a document requires the instantiation of a XadesSigner as in

Code Block 6.3 where a BES profile is being used.

```
public XadesSigner XadesBESSign() throws XadesProfileResolutionException{

    KeyingDataProvider kdp = getPKCS11rovider(new certSelector(signeeUN),
        new passwordProvider(), new keyEntryProvider());
    XadesBesSigningProfile xbp= new XadesBesSigningProfile(kdp);
    xbp.withSignaturePropertiesProvider(signatureProperties.class)
       .withAlgorithmsProvider(myProvider.class);

return xbp.newSigner();
}
```

Code Block 6.3: Xades-BES profile

In order to sign a document, we are required to define which kind of Xades signature we intend to perform. We defined a Enveloped signature to provide self-contained XML documents.

```
public void signFile(Document doc) throws XAdES4jException, ProviderException {
    XadesSigner xs = XadesBESSign();
    Node n = doc.getDocumentElement();
    SignedDataObjects dataObjs = new SignedDataObjects(
        new EnvelopedXmlObject(n)).withCommitmentType(
                AllDataObjsCommitmentTypeProperty.proofOfApproval());

    xs.sign(dataObjs,doc);
    sendTo(postTO,doc);
}
```

Code Block 6.4: Signing a Document

Signed documents as observed in Code Block 6.4 are sent to the URL defined in the postTO applet parameter.

```
public void sendTo(String postTO, Document doc)
   throws IOException,ParseException,TransformerException {
            ByteArrayInputStream inputStream = null;
            HttpClient httpclient = new DefaultHttpClient();
            httpclient.getParams().setParameter(CoreProtocolPNames.PROTOCOL_VERSION,
                HttpVersion.HTTP_1_1);
            HttpPost httppost = new HttpPost(postTO);
            MultipartEntity mpEntity = new MultipartEntity();
            Writer writer = new StringWriter();
            TransformerFactory tf = TransformerFactory.newInstance();
            tf.newTransformer().transform(new DOMSource(doc), new StreamResult(writer));
            inputStream = new ByteArrayInputStream(writer.toString().getBytes("UTF-8"));
            ContentBody cbody = new InputStreamBody(inputStream,"file");
            mpEntity.addPart("File", cbody);
            httppost.setEntity(mpEntity);
            String response = EntityUtils.toString( httpclient.execute( httppost ).getEntity(), "
                UTF-8" );
            httpclient.getConnectionManager().shutdown();
}
```

Code Block 6.5: Uploading a document

Smartcards require middleware to operate,but in order to avoid problems such as requiring the user to have previously installed the smartcard's middleware, the applet can identify the operative system and its architecture obtaining the middleware online without the user's intervention. Currently, the supported cards are both the Portuguese National identity card and the university card.

6.3 Signserver

PrimeKey's signserver did not support XaDes signatures at the time the implementation started. In order to solve this issue, a XadeS signer module which can perform HSM-supported XaDes signatures with different profiles was developed and subsequently adopted by Primekey **??**.

Signserver exposes three different kinds of processable services:

- Signers

  Responsible for processing requested data and performing signatures

- Validation Services

  Service that provides the capability to validate a signed document's certificate against a set of backed issuers.

- Group key service framework

  Service that manages and distributes group keys for different applications.

Our solution requires implementation of both the signer service and validation service in order to provide an effective method for signing documents as well as validating signed documents.

The creation of a service in signserver is as simple as extending the base signer class.

```
public class XadesSigner extends BaseSigner{
  @Override
    public void init(int signerId, WorkerConfig config,
            WorkerContext workerContext, EntityManager workerEntityManager) {
    }
  @Override
    public ProcessResponse processData(ProcessRequest signRequest, RequestContext
        requestContext) throws IllegalRequestException, CryptoTokenOfflineException,
        SignServerException {


    }
}
```

Code Block 6.6: Extending Base Signer

In order to perform XaDes signatures with different profiles the xades4j library was selected due to its open source nature and support of different XadeS profiles.

Configuration follows the same principle as with other signserver services. When using a generic signer supporting XaDes,the configuration of the profile to be used in the signature process is required and the following properties are recognized by the signer:

- XADESFORM

  This property defines which XaDes profile is used by the signer instance to perform the signature. Currently this property can have the following values BES,C,EPES and T.

- TSA_URL

  Using a profile that requires a timestamp to be obtained, makes this property required. The TSA_URL string is used as a timestamp URL.

- TSA_USERNAME

  This property defines the username to be used when TSA authentication is required. It defaults to non-authenticated if the property is not set.

- TSA_PASSWORD

  This property sets the passwords to be used with the username if set. Defaults to null otherwise.

- MARKLOGIC_URL

  The MARKLOGIC_URL property when set, provides signed documents archival in a mark logic database. When the property is not set, the signer does not archive the document and simply returns the signed version to the requester.

- MARKLOGIC_PORT

  The Marklogic's server port can be set with this property. Required if MARKLOGIC_URL is set.

- MARKLOGIC_USER

  The property establishes the username to be used in the database authentication process. Required if MARKLOGIC_URL is set.

- MARKLOGIC_PASSWORD

  The password to be paired with the username. Required if MARKLOGIC_URL is set.

As previously described, the signers take multiple configurable parameters defined in a property file.

```
# Example Xades Signer configuration

GLOB.WORKERGENID1.CLASSPATH = org.signserver.module.xadessinger.XadesSigner
#GLOB.WORKERGENID1.SIGNERTOKEN.CLASSPATH = org.signserver.server.cryptotokens.
    SoftCryptoToken

## General properties

WORKERGENID1.NAME=XadesSigner
WORKERGENID1.AUTHTYPE=NOAUTH

#XADESFORM can be BES,C,EPES,T
WORKERGENID1.XADESFORM=T
#(required if implicit in xades profile)
WORKERGENID1.TSA_URL=http://localhost:8080/signserver/tsa?workerName\=TSA

#optional
#WORKERGENID1.TSA_USERNAME=
#optional
#WORKERGENID1.TSA_PASSWORD=

#optional
#WORKERGENID1.MARKLOGIC_URL=
#WORKERGENID1.MARKLOGIC_USER=
#WORKERGENID1.MARKLOGIC_PASSWORD=
```

Code Block 6.7: Sample Configuration

Although a generic implementation was developed in order to contribute with patches to the sign server project, a specific signer and validator aware of our dynamic trust delegation chains is required. While performing an institutional signature committing to the validity of a document, the verification of document's authorship while establishing the author's permission to produce the document is of extreme importance. An approved and institutionally signed document cannot be refuted, and it is bound to the institution's reputation.

In order to facilitate this verification, two validators were developed for the documents:

- Intermediate Document Validator

  It validates documents that do not contain institutional signatures.

  The document's integrity is verified using the signatures present in the document and using the document representing the delegation chain. The validator recursively verifies the delegations until the main trust anchor is reached.

  Documents which do not provide at least a signature from a trusted signee are rejected as invalid.

- Final Document Validator

  Final documents are required to contain institutional signatures.

  The document's signatures are validated using the intermediate documents service and, if a positive response is obtained, a signature containing the institution's certificate in the document is searched before establishing the document's validity.

The validators are extended from the BaseValidator class as described in Code Block 6.8

```
class XadesValidator extends BaseValidator {

    public ProcessResponse processData(ProcessRequest signRequest, RequestContext
        requestContext) throws IllegalRequestException, CryptoTokenOfflineException,
        SignServerException {


    }

    private GenericValidationResponse validate(final int requestId, byte[] data) throws
        SignServerException {

    }
```

Code Block 6.8: Extending Base Signer

The full validator responsible for verifying if a document was accepted by the institution (contains a valid institutional certificate) is available in Appendix B.

6.4 Differences with University of Murcia Framework

In subsection 4.1.6 we identified some problems with the architecture of the University of Murcia solution. In order to guarantee that an author's ability to produce a given document is verifiable in long-term archives, instead of relying on volatile permissions recorded in the information system, we designed a method for delegating trust for each user and role. By using signed documents guaranteeing long-term integrity and verifiability we provide an effective mechanism which also allows authors to delegate tasks while retaining audit capabilities.

Another problem identified in the University of Murcia framework pertains to the absence of a trusted third-party and the impossibility to have mutual non-repudiation. To address this concern, our proposal includes a timestamp obtained from an external entity. An external timestamp, anchors a third-party to the document's production, guaranteeing non-repudiation of its existence.

A specific implementation concern is the method for producing digital signatures client-side. Using Microsoft deprecated CAPICOM and highly browser-specific implementations may lead to a set of problems. Providing a PKCS#11-based cross-platform solution based on Java Applets seems a better fit, smartcard middleware could be loaded dynamically from online sources, and the maintenance of a single Applet is less error-prone.

Chapter 7

Conclusions

The development in the information system is critical, prone to bureaucracy and, unfortunately, fragmented, resulting in a longer project for completion. To solve these problems, a similar system following the document's workflow was created as a proof of concept, providing a mechanism to test delegation attribution, student evaluation and document signatures, as well as institutional signatures, document archive and validators for generated documents.

While the objective of integrating the architectural changes to the current information system was not concluded, a few problems that require changes in the university itself were identified. Although it was reasonably assumed that the actors who perform given tasks are well identified, some roles such as who can sign a given document are not well defined (as when directors can sign grades for invited professors). Each faculty inside the university may delegate the tasks to different persons and while our delegation chain could cope with delegations, establishing trusted anchors would require some effort). Also, in some cases, documents can be signed by a superior, thus bypassing the actual document creator. Although we can provide an inheritance mechanism to our delegation chain, it subverts the non-repudiation property of document's authorship,which invalidates the general scheme for different faculties (inheriting trust due to group roles).

The proof of concept was comprised by a set of web services and front-end controllers which emulate the workflow of the grading process. A trusted anchor could delegate evaluation tasks to professors who then could evaluate students by producing the grading documents, that, after being institutionally signed, would be archived in the XML database.

The produced documents were validated and inspected to ascertain conformity with the specification.

Different attacks were performed as to establish the information system ability to detect manipulated documents, as well as revoked trust delegations and expired certificates.

Chapter 8

Future Work

During the research and development of this architecture, a new method for document dematerialization was drafted in order to facilitate information sharing across institutions. While XML provides the most efficient way to share information across information systems, legacy systems must be taken into account. XML signatures are not easily validated outside the scope of an information system and they do not provide a visual representation per se.

To solve the problem of information sharing with institutions that still rely on older mechanisms, providing an embedded XML inside an institutionally signed PDF document with a visual representation of the given XML can support an easier offline document verification in a legacy environment. While this draft was published [7] the supporting mechanism for such a schema remain unimplemented and could represent an important step into adoption of different trusted document models across institutions.

Appendix A

XaDeS Signing Module

```
package org.signserver.module.xadessigner;

import com.marklogic.client.DatabaseClient;
import com.marklogic.client.DatabaseClientFactory;
import com.marklogic.client.document.XMLDocumentManager;
import com.marklogic.client.io.DOMHandle;
import com.marklogic.client.io.XMLStreamReaderHandle;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.security.cert.X509Certificate;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.logging.Level;
import javax.persistence.EntityManager;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.stream.XMLStreamReader;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.signserver.common.CryptoTokenOfflineException;
import org.signserver.common.IllegalRequestException;
import org.signserver.common.ProcessRequest;
```

```
import org.signserver.common.ProcessResponse;
import org.signserver.common.RequestContext;
import org.signserver.common.SignServerException;
import org.signserver.server.signers.BaseSigner;
import org.apache.log4j.Logger;
import org.bouncycastle.util.encoders.Hex;
import org.ejbca.util.CertTools;
import org.signserver.common.ArchiveData;
import org.signserver.common.GenericServletRequest;
import org.signserver.common.GenericServletResponse;
import org.signserver.common.GenericSignRequest;
import org.signserver.common.GenericSignResponse;
import org.signserver.common.ISignRequest;
import org.signserver.common.WorkerConfig;
import org.signserver.server.WorkerContext;
import org.signserver.server.cryptotokens.ICryptoToken;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.xml.sax.SAXException;
import xades4j.XAdES4jException;
import xades4j.production.Enveloped;
import xades4j.production.EnvelopedXmlObject;
import xades4j.production.SignedDataObjects;
import xades4j.production.XadesBesSigningProfile;
import xades4j.production.XadesCSigningProfile;
import xades4j.production.XadesSigningProfile;
import xades4j.production.XadesTSigningProfile;
import xades4j.properties.AllDataObjsCommitmentTypeProperty;
import xades4j.providers.KeyingDataProvider;
import xades4j.providers.impl.DirectKeyingDataProvider;
import xades4j.utils.XadesProfileResolutionException;
import xades4j.providers.impl.TimeStampProvider;

/**
 *
 * @author Luis Maia <lmaia@dcc.fc.up.pt>
 */
public class XadesSigner extends BaseSigner{
    private static final Logger LOG = Logger.getLogger(XadesSigner.class);;
    static String XADESFORMDEFAULT="BES";
    static String XADESFORM="XADESFORM";
    static String TSA_URL="TSA_URL";
    static String TSA_USERNAME="TSA_USERNAME";
    static String TSA_PASSWORD="TSA_PASSWORD";
```

```
    public static final String DEFAULT_ARCHIVETODISK_FILENAME_PATTERN = "${
        WORKERID}-${REQUESTID}-${DATE:HHmmssSSS}.pdf";




    /* Supported Xades Profiles*/
    public enum Profiles {
    BES,
    C,
    EPES,
    T
}


    @Override
    public void init(int signerId, WorkerConfig config,
            WorkerContext workerContext, EntityManager workerEntityManager) {
        super.init(signerId, config, workerContext, workerEntityManager);
    }
    @Override
    public ProcessResponse processData(ProcessRequest signRequest,   RequestContext
        requestContext) throws IllegalRequestException, CryptoTokenOfflineException,
        SignServerException {
        ProcessResponse signResponse;
          ISignRequest sReq = (ISignRequest) signRequest;

          // Check that the request contains a valid GenericSignRequest object with a byte[].
          if (!(signRequest instanceof GenericSignRequest)) {
              throw new IllegalRequestException("Recieved request wasn't a expected
                  GenericSignRequest.");
          }
          if (!(sReq.getRequestData() instanceof byte[])) {
              throw new IllegalRequestException("Recieved request data wasn't a expected byte[].
                  ");
          }

          byte[] data = (byte[]) sReq.getRequestData();
          /*SHA1 fingerprint for the request returned*/
          byte[] fpbytes = CertTools.generateSHA1Fingerprint(data);
          String fp = new String(Hex.encode(fpbytes));
          XadesSignerParameters params = new XadesSignerParameters(workerId, config);
          xades4j.production.XadesSigner xs=null;
          Document doc = null;
        try {
```

```
        xs = XadesSign(params);
        DocumentBuilderFactory factory =  DocumentBuilderFactory.newInstance();
        factory.setNamespaceAware(true);
        DocumentBuilder builder = factory.newDocumentBuilder();
        doc = builder.parse(new ByteArrayInputStream(data));
        if(!validateSignees(doc))
            {
                    throw new IllegalRequestException("Invalid Signer in document.");
            }
        Node n = doc.getDocumentElement();
        SignedDataObjects dataObjs = new SignedDataObjects(new EnvelopedXmlObject(n))
            .withCommitmentType(AllDataObjsCommitmentTypeProperty.proofOfApproval()
                );
         xs.sign(dataObjs,doc);


    } catch (SAXException ex) {
        java.util.logging.Logger.getLogger(XadesSigner.class.getName()).log(Level.SEVERE,
            null, ex);
    } catch (IOException ex) {
        java.util.logging.Logger.getLogger(XadesSigner.class.getName()).log(Level.SEVERE,
            null, ex);
    } catch (ParserConfigurationException ex) {
          throw new SignServerException("Document parsing error", ex);
    } catch (XadesProfileResolutionException ex) {
           throw new SignServerException("Exception in Xades Profile Resolution", ex);
    } catch (XAdES4jException ex) {
        java.util.logging.Logger.getLogger(XadesSigner.class.getName()).log(Level.SEVERE,
            null, ex);
    }


    ByteArrayOutputStream bout = new ByteArrayOutputStream();

    TransformerFactory tf = TransformerFactory.newInstance();
    Transformer trans;
    try {
        trans = tf.newTransformer();
        trans.transform(new DOMSource(doc), new StreamResult(bout));
         /*Do the archiving*/
         archive(doc,params.getMarkLogicParameters(),sReq.getRequestID());

    } catch (TransformerException ex) {
        java.util.logging.Logger.getLogger(XadesSigner.class.getName()).log(Level.SEVERE,
            null, ex);
```

```
        }



        final byte[] signedbytes = bout.toByteArray();
        if (signRequest instanceof GenericServletRequest) {
            signResponse = new GenericServletResponse(sReq.getRequestID(), signedbytes,
                getSigningCertificate(), fp, new ArchiveData(signedbytes), "text/xml");
        } else {
            signResponse = new GenericSignResponse(sReq.getRequestID(), signedbytes,
                getSigningCertificate(), fp, new ArchiveData(signedbytes));
        }
        return signResponse;

    }

      public xades4j.production.XadesSigner XadesSign(XadesSignerParameters params) throws
          XadesProfileResolutionException, SignServerException{
        KeyingDataProvider kdp=null;

        try {

        kdp = new DirectKeyingDataProvider((X509Certificate)this.getSigningCertificate(), this.
            getCryptoToken().getPrivateKey(ICryptoToken.PURPOSE_SIGN));
        } catch (CryptoTokenOfflineException ex) {
            java.util.logging.Logger.getLogger(XadesSigner.class.getName()).log(Level.SEVERE,
                null, ex);
        }

        Profiles config_profile = Profiles.valueOf(params.getXadesForm());
        XadesSigningProfile xsp=null;
        switch(config_profile){
            case C:
                //unimplemented yet
            case EPES:
                //unimplemented yet
            case BES:
                xsp = new XadesBesSigningProfile(kdp);
                break;
            case T:
                TSAParameters tsaParameters = params.getTSAParameters();
                if(tsaParameters==null){
                    throw new SignServerException("TSA URL is required to use the "+
                        config_profile.toString()+" Profile");
```

```
                }
                xsp = new XadesTSigningProfile(kdp)
                        .withTimeStampTokenProvider(TimeStampProvider.class)
                        .withBinding(TSAParameters.class, params.getTSAParameters());
                break;
            default:
                throw new XadesProfileResolutionException("Unknown Xades Profile", null);
        }

        return (xades4j.production.XadesSigner) xsp.newSigner();

    }

  private void archive(Document doc,MarkLogicParameters params,int requestID) {
        if(params!=null){
            DatabaseClient dbclient = DatabaseClientFactory.newClient(params.getURL(),
                 params.getPort(), params.getUsername(), params.getPassword(),
                 DatabaseClientFactory.Authentication.DIGEST);
            XMLDocumentManager docMgr = dbclient.newXMLDocumentManager();
            DOMHandle domHandle = new DOMHandle();
            domHandle.set(doc);

            /*Create a docID */
            final SimpleDateFormat sdf = new SimpleDateFormat("yyyy_MM_dd");
              String data = sdf.format(new Date());
            String URI = "/XadesSigner/"+workerId+"-"+requestID+"-"+data;
            docMgr.write(URI,domHandle);
            dbclient.release();

        }
    }

}
```

Appendix B

XaDeS validator Module

```
package org.signserver.module.xadesvalidator;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.cert.CertificateException;
import java.util.logging.Level;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.apache.log4j.Logger;
import org.apache.xml.security.utils.Constants;
import org.bouncycastle.asn1.x500.RDN;
import org.bouncycastle.asn1.x500.X500Name;
import org.bouncycastle.asn1.x500.style.BCStyle;
import org.bouncycastle.cert.jcajce.JcaX509CertificateHolder;

import org.signserver.common.CryptoTokenOfflineException;
import org.signserver.common.GenericServletRequest;
import org.signserver.common.GenericValidationRequest;
import org.signserver.common.GenericValidationResponse;
import org.signserver.common.IValidationRequest;
import org.signserver.common.IllegalRequestException;
import org.signserver.common.ProcessRequest;
import org.signserver.common.ProcessResponse;
import org.signserver.common.RequestContext;
```

```
import org.signserver.common.SignServerException;
import org.signserver.server.validators.BaseValidator;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
import xades4j.XAdES4jException;
import xades4j.providers.CertificateValidationProvider;
import xades4j.providers.impl.PKIXCertificateValidationProvider;
import xades4j.verification.XAdESVerificationResult;
import xades4j.verification.XadesVerificationProfile;
import xades4j.verification.XadesVerifier;

/**
 * lmaia@dcc.fc.up.pt
 */
public class XadesValidator extends BaseValidator {

    /** Logger. */
    private static final Logger LOG = Logger.getLogger(XadesValidator.class);



    public ProcessResponse processData(ProcessRequest signRequest, RequestContext
        requestContext) throws IllegalRequestException, CryptoTokenOfflineException,
        SignServerException {

        // Check that the request contains a valid GenericSignRequest object with a byte[].
        if (!(signRequest instanceof GenericValidationRequest)) {
            throw new IllegalRequestException("Recieved request wasn't a expected
                GenericValidationRequest.");
        }
        IValidationRequest sReq = (IValidationRequest) signRequest;

        if (!(sReq.getRequestData() instanceof byte[])) {
            throw new IllegalRequestException("Recieved request data wasn't a expected byte[].");
        }
        if (signRequest instanceof GenericServletRequest) {
            throw new IllegalArgumentException("GenericServletRequest not yet supported");
        }

        byte[] data = (byte[]) sReq.getRequestData();
        GenericValidationResponse response = validate(sReq.getRequestID(), data);
        return response;
    }
```

```
private GenericValidationResponse validate(final int requestId, byte[] data) throws
    SignServerException {

    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    dbf.setNamespaceAware(true);

    try {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        factory.setNamespaceAware(true);
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document doc = builder.parse(new ByteArrayInputStream(data));

        KeyStore truststore = KeyStore.getInstance(KeyStore.getDefaultType());
        char[] password = getConfig().getProperty("TRUSTSTOREPASSWORD").
            toCharArray();
            java.io.FileInputStream fis =
                new java.io.FileInputStream(getConfig().getProperty("TRUSTSTOREPATH")
                    );
        truststore.load(fis, password);
        fis.close();
        CertificateValidationProvider cvp = new PKIXCertificateValidationProvider(truststore
            , false);

        XadesVerificationProfile p = new XadesVerificationProfile(cvp);

        XadesVerifier xv = p.newVerifier();



        NodeList nl = doc.getElementsByTagNameNS(Constants.SignatureSpecNS, Constants.
            _TAG_SIGNATURE);

        if(nl.getLength()==0){
            return new GenericValidationResponse(requestId, false);
        }
        String signees = "";
        boolean isSignedbyServer = false;
        for(int i=0;i<nl.getLength();i++){
            Element sigElement= (Element) nl.item(i);
            XAdESVerificationResult r = xv.verify(sigElement,null);


          X500Name x500name = new JcaX509CertificateHolder(r.getValidationCertificate()).
              getSubject();
```

```
                RDN[] rdn = x500name.getRDNs(BCStyle.CN);
                String cn = rdn[0].getFirst().getValue().toString();
                if(cn.equalsIgnoreCase("Signature Service")) //since we are "pinning" certs this is
                    OUR cert
                {
                    isSignedbyServer=true;
                }
            }

            return new GenericValidationResponse(requestId, isSignedbyServer);



        } catch (NoSuchAlgorithmException ex) {
            java.util.logging.Logger.getLogger(XadesValidator.class.getName()).log(Level.SEVERE,
                null, ex);
        } catch (CertificateException ex) {
            java.util.logging.Logger.getLogger(XadesValidator.class.getName()).log(Level.SEVERE,
                null, ex);
        } catch (KeyStoreException ex) {
            java.util.logging.Logger.getLogger(XadesValidator.class.getName()).log(Level.SEVERE,
                null, ex);
        } catch (XAdES4jException ex) {
            java.util.logging.Logger.getLogger(XadesValidator.class.getName()).log(Level.SEVERE,
                null, ex);
        }catch (NoSuchProviderException ex) {
            java.util.logging.Logger.getLogger(XadesValidator.class.getName()).log(Level.SEVERE,
                null, ex);
        } catch (SAXException ex) {
            java.util.logging.Logger.getLogger(XadesVerifier.class.getName()).log(Level.SEVERE,
                null, ex);
        } catch (IOException ex) {
            java.util.logging.Logger.getLogger(XadesVerifier.class.getName()).log(Level.SEVERE,
                null, ex);
        } catch (ParserConfigurationException ex) {
            java.util.logging.Logger.getLogger(XadesVerifier.class.getName()).log(Level.SEVERE,
                null, ex);
        }
        //any error ocurring means the document is NOT properly validated (false)
        return new GenericValidationResponse(requestId, false);

        }

    }
```

References

[1] J. Barney, "Firm resources and sustained competitive advantage," *Journal of Management*, vol. 17, pp. 99–120, 1991. 1

[2] R. Grewal, J. A. Dearden, and G. L. Llilien, "The university rankings game," *The American Statistician*, vol. 62, no. 3, 2008. 1

[3] X. Luo and M. Warkentin, "Assessment of information security spending and costs of failure.," in *Proceedings of 2004 ISOneWorld International Conference, Las Vegas, NV*, 2004. 1

[4] R. Cummings, "The evolution of information assurance," *Computer*, vol. 35, no. 12, pp. 65–72, 2002. 1

[5] K. Dauch, A. Hovak, and R. Nestler, "Information Assurance Using a Defense In-Depth Strategy," in *CATCH 2009: CYBERSECURITY APPLICATIONS AND TECHNOLOGY CONFERENCE FOR HOMELAND SECURITY, PROCEEDINGS*, pp. 267–272, Dept Homeland Security, 2009. Cyber Security Applications and Technology Conference for Homeland Security, Washington, DC, MAR 03-04, 2009. 1

[6] L. Maia and M. Correia, "Java jca/jce programming in android with sd smart cards," in *Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on*, pp. 1–6, 2012. 6

[7] L. A. Maia, L. M. Valente, M. E. Correia, and L. M. Ribeiro, "Trusted information across student information systems," 2013. 6, 9, 10, 60

[8] Primekey, "Primekey's signserver." `http://www.signserver.org/`, 2013. 6, 32

[9] M. L. Markus Kilås, Luis A. Maia, "Primekey bug tracker." `https://jira.primekey.se/browse/DSS-12`, Nov. 2013. 6

[10] L. M. Ribeiro, G. David, A. Azevedo, and J. M. dos Santos, "Developing an information system at the engineering faculty of porto university," *Proceedings of the EUNIS 97-European Cooperation in Higher Education Information Systems*, 1997. 8

[11] G. David and L. M. Ribeiro, "Getting management support from an university information system," *EUNIS 99 Information Technology Shaping European Universities*, 1999. 8

[12] J. Nickell, *Detecting Forgery: Forensic Investigation of Documents*. University Press of Kentucky, 2005. 9

[13] C. S. et al?, "Guidelines on a european learner mobility model." `http://wiki.teria.no/display/EuropeanLearnerMobility/Guidelines+on+European+Learner+Mobility`. 10

[14] C. o. E. European Commission and UNESCO/CEPES, "Outline structure for the diploma supplement." `http://ec.europa.eu/education/lifelong-learning-policy/doc/ds/ds_en.pdf`. 10

[15] C. O. T. E. COMMUNITIES, "Report from the commission to the european parliament and the council on the first evaluation of the europass initiative." `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2008:0427:FIN:EN:PDF`, July 2008. 10

[16] F. D. Schoorman, R. C. Mayer, and J. H. Davis, "An integrative model of organizational trust: Past, present, and future," *Academy of Management review*, vol. 32, no. 2, pp. 344–354, 2007. 10

[17] H. Li and M. Singhal, "Trust management in distributed systems," *Computer*, vol. 40, pp. 45–53, Feb. 2007. 11

[18] B. Christianson and W. S. Harbison, "Why isn't trust transitive?," in *Proceedings of the International Workshop on Security Protocols*, (London, UK, UK), pp. 171–176, Springer-Verlag, 1997. 11

[19] A. Jøsang and S. Pope, "Semantic constraints for trust transitivity," in *Proceedings of the 2Nd Asia-Pacific Conference on Conceptual Modelling - Volume 43*, APCCM '05, (Darlinghurst, Australia, Australia), pp. 59–68, Australian Computer Society, Inc., 2005. 11

[20] E. Barka and R. Sandhu, "Framework for role-based delegation models," in *Computer Security Applications, 2000. ACSAC '00. 16th Annual Conference*, pp. 168–176, 2000. 12

[21] L. Zhang, G.-J. Ahn, and B.-T. Chu, "A rule-based framework for role-based delegation and revocation," *ACM Trans. Inf. Syst. Secur.*, vol. 6, pp. 404–441, Aug. 2003. 12

[22] Y. Ding, P. Horster, and H. Petersen, "A new approach for delegation using hierarchical delegation tokens.," in *Communications and Multimedia Security*, pp. 128–143, Citeseer, 1996. 12

[23] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644–654, 1976. 16

[24] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Rfc 5280: Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile," *URL: http://www. ietf. org/rfc/rfc5280. txt*, 2008. 18

[25] SEMA/UMIC/AMA, "Projecto cartão de cidadão - especificações leitor base." `http://www.cartaodecidadao.pt/images/stories/especificacoes%20-%20leitores%20base_v1.0.pdf`, Nov. 2013. 20, 21

[26] L. Valente, "Mecanismos seguros para o auto-aprovisionamento de certificados do Cartão U.Porto," Master's thesis, Faculty of Science, University of Porto, December, 2012. 21

[27] Mastercard, "Paypass – m/chip 4 - card technical specification," 2008. Restricted document. 21

[28] "Marklogic." `http://www.marklogic.com/`, Nov. 2013. 32

[29] D. Sánchez-Martínez, I. Marín-López, J. Fabre-Cascales, T. Jiménez-García, and A. F. Gómez-Skarmeta, "THE BUILDING OF A UBIQUITOUS GOVERNMENT IN THE UNIVERSITY OF MURCIA," 36

[30] MICROSOFT, "Capicom reference." `http://msdn.microsoft.com/en-us/library/windows/desktop/aa375732%28v=vs.85%29.aspx`, Nov. 2013. 38

[31] T. Coffey and P. Saidha, "Non-repudiation with mandatory proof of receipt," *Computer Communication Review*, vol. 26, pp. 6–17, 1996. 40