Optimized Public Key Infrastructure

A PKI to Support Efficient Document's Signatures

Martín Augusto Gagliotti Vigil*, Ricardo Felipe Custódio*, Nelson da Silva*, Ricardo Moraes*†
* Computer Security Laboratory (LabSEC) - Federal University of Santa Catarina, Brazil
† IT-Instituto de Telecomunicações, University of Aveiro, Portugal
{vigil, custodio, nelson, rmoraes}@inf.ufsc.br

ABSTRACT

Optimized Public Key Infrastructures are traditional PKI in which end users may optimize the signatures of their documents, replacing the signer's validation data with Optimized Certificates (OC). OCs carry the signer's identification and public key, but are issued for a specific time, i.e., fields *notBefore* and *notAfter* have the same value, thus there are no reasons to revoke them. The OC's certification path is supposed to be shorter and uses Micali's revocation scheme. Furthermore, OCs include signed document's hash-codes, working also as time-stamps. Therefore, OCs are useful to replace signed document's validation data by one smaller and easier to verify. Finally, when OCs become invalid due to cryptographic algorithm weakness and limits in the validity periods of their certificate chains, they can be easily replaced by new ones, thus this proposal is suitable for efficient long term archiving.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

X.509 standards

Keywords

Optimized certificates, certification authority

1. INTRODUCTION

Over the last few years the topic of electronic commerce has played an import role in business. On the one hand, there is an attractive scenario where companies and institutions are joining to look for new job opportunities. On the other, every day we read about companies suffering millions of dollars in losses due to security breaches. Additionally, a number of government sectors and other institutions are storing and sharing information on digital media, which highlights the security need in these emerging applications.

Within this context, a public key cryptography approach was initially proposed by Loren M. Kohnfelder [1] and, currently, a Public Key Infrastructure (PKI) is widely used to guarantee the desired security properties, such as confidence, integrity, authentication and no-repudiation. However, even in emerging applications with some challenges in implementing security mechanisms, it has not been used due to complex reasons, for example: in wireless industrial networks, wireless sensor networks and in embedded systems with energy and processing restrictions [2, 3, 4].

One of the most relevant complexity examples is the validation of digital signatures whose accomplishment requires several steps. Firstly, it is necessary to verify the signature against the signer's public key that is embedded in the signer's certificate. Secondly, that certificate should be validated through the X509 certificate validation algorithm [5], which consists of discovering a certification path between the signer's certificate and a previously trusted certificate, for instance a Root Certificate Authority (Root CA). Afterwards, when a certification path is discovered, all certificates in this path must also be validated: attesting their signatures, validity period, revocation status and policy restrictions. Then, if the certification path length is long the load of the verifier becomes overly heavy.

Moreover, signed documents often incorporate time-stamps [6], which are useful to certify the date and time that a digital signature was created. A time-stamp is a particular kind of signed document, demanding validations of signature and the time-stamp authority's certificate. Thus, it is also a source of complexity. Finally, another drawback of PKI for some application areas is the certificate revocation process. Usually, there are two alternatives to represent the revocation status in an electronic document: incorporating the whole revocation status data or only its references. In the former case, all information about revocation is embedded in the signed document which gets bigger but easier to verify. In the latter, a minimum set of data is included in the document. However, the verification task is more inefficient than the first case since external queries are needed to retrieve missing data. Therefore, there is a trade-off between efficient signature's format and validation.

This paper proposes a simple and effective solution to overcome some of the previously described sources of complexity in a Public Key Infrastructure. It is propose an "Optimized Public Key Infrastructure - OPKI", which issues a new kind of X509v3 certificate, named Optimized Certifi-

cates (OC) - whose main goal is to make the validation and format of an existing digital signature more efficient [7]. The main idea of the OPKI consists of issuing OC for a specific instant of time. The main contributions of this work are:

- time-stamp functionality embedded in OCs;
- an efficient approach to reduce digital signature storage requirements and its validation processing as well as bandwidth costs within a given domain;
- relatively fixed requirements and lightweight method for verifying digital signatures on a long term basis;
- deployment of Novomodo validity proofs in order to enhance the OCCA's relative time-stamp security.

The remainder of this paper is organized as follows. Section 2 presents a review of the validation and format of signed documents, which is important to comprehend the trade-off between efficient verification and the amount of validation data embedded in a signed document. Section 3 mentions some proposals of new PKI schemes as means to solve the signature issues presented in Section 2. Those works have been used as a basis for OC's concepts. Section 4 highlights the proposed concepts of the Optimized PKI and its benefits. Section 5 presents the Optimized PKI in practice and its trust relationship with users from a domain. Section 6 deals with the validation of digital signatures using OCs. Section 7 presents a comparison between OCs and conventional X509 certificates. Finally, in Section 8, some conclusions and future work are drawn.

2. BACKGROUND

In order to understand the optimization process proposed in this work, the basics of the digital signature, highlighting both validation and format of signed documents, are described. Furthermore, a real example of these issues is exposed.

2.1 Signature Validation

As a means to attest a document's signature a verifier should validate the digital signature and also the signer's certificate. The first validation consists of a cryptographic verification using the signer's public key that assures integrity and authenticity of the signature. The second validation is performed to guarantee that the signer's certificate was valid when the signature was created, which is achieved throughout construction and validation of a certification path [5, 8].

The objective of the certification path construction process is to find out one or more sequence of certificates that links the signer's certificate to a trust anchor: a certificate of an entity that was deemed trustworthy by some adopted policy. Finally, once sequences are found they have to be validated. Thus, for each of the sequence's certificates the following requirements needed to be assured:

- signature integrity and authenticity using certificate's issuer public key;
- the validity period includes the document's signature creation time;
- the certificates have not been revoked before the document's signature creation time;
- subject distinguished names follows any existing name constraints;
- the certificate follows all certification policies stated by its PKI.

Revocation status is often acquired from certificate's issuer and this operation usually reduces the performance of the signature validation process. The two most popular revocation schemes are *Certificate Revocation List* (CRL) [9] and *Online Certificate Status Protocol* (OCSP) [10]. The former is a signed and time-stamped list in which a CA informs all its issued certificates that have been compromised. This list is published in a distribution point. The latter is an online service provided by a trusted entity to which a user can ask about the revocation status of a certificate. That entity finds out the inquired revocation status and answers the user through a time-stamped and signed message.

One of the shortcomings of CRLs is that they can become awfully long, what implies a poor scalability, which influences update, bandwidth and storage costs. On the other hand, even though OCSP reduces user computational effort, it requires the user to perform external queries to OCSP server, which may be unreachable. Furthermore, CRL and OCSP approaches use signed content whose signature should be verified, which makes those revocation schemes more complex.

Finally, it can be concluded that the certification path validation is a high cost procedure whose required effort is proportional to the certification path length. Therefore, the ideal certification path should be as short as possible, for example, a hierarchic PKI of one level in which a Root CA issues end user certificates. However, this is not a fair solution since there would be operational issues, such as Root CA overload and security.

2.2 Digital Signature Format

In an effort to set up a signature format standard, some technical specifications have been standardized, such as *PKCS#7* [11], *CMS* [12], *CAdES* [13, 14] and *XAdES* [15]. This work considers *CAdES* since it is a recent standard and focuses on its relation with the validation data useful to verify an electronic signature: certificates, CRLs, certificate status information (OCSP) and time-stamps [14]. Additionally, signature policies are out of the scope of this paper, thus *CAdES-EPES* has not been cited as well as examples do not mention the signature policy identification.

First, there is *CAdES-BES* format, which carries no validation data (Figure 1). It is the minimum format for an electronic signature.

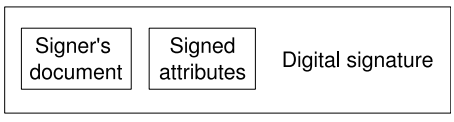

Figure 1: *CAdES-BES* and absence of validation data.

In *CAdES-BES*, a signature is accepted as valid only until a certificate in the signer's certificate chain expires or is revoked. To overcome this restriction, there is *CAdES-T* format in which a time-stamp can be included as shown in Figure 2.

Figure 2: *CAdES-T* format.

As a conventional signed document can be widely distributed and there is no embedded reference to validation data in both *CAdES-BES* and *CAdES-T*, the previous approaches often becomes useless in many scenarios. Thus, one could choose two alternative formats: *CAdES-C* or *CAdES-X Long*, which are respectively illustrated in Figures 3 and 4.

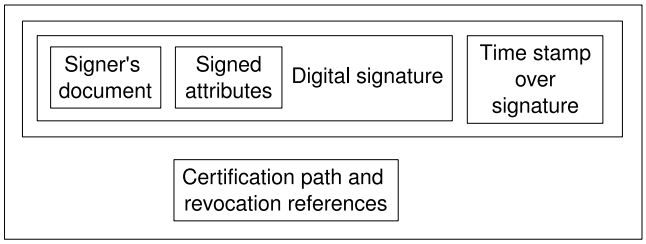

Figure 3: *CAdES-C* format.

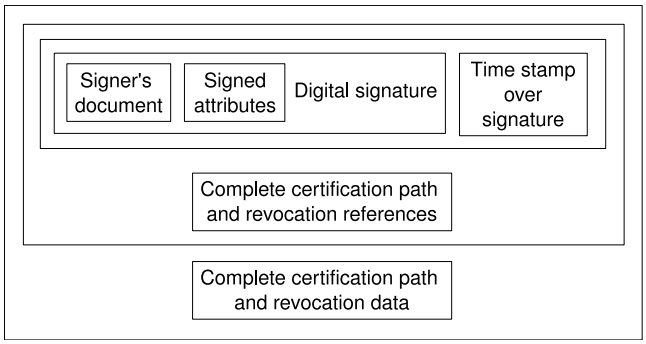

Figure 4: *CAdES-X Long* format.

The difference between these two formats is the inclusion of complete validation data, which is only included in *CAdES-X Long*. Both alternatives have advantages and disadvantages related to efficient means for storing and validating a signed document.

Taking *CAdES-C* as an alternative, one can have a signed document whose size is as small as possible since validation data is not included. Thus, this approach saves storage space. However, a means is required, for example a public directory, through which the signers' validation data is available for whoever wants to verify a signature at any time and from anywhere. Therefore, that scenario demands high network availability otherwise signers could neither verify their certificates before signing nor could verifiers check signed documents. Moreover, significant bandwidth usage and cost due to signatures verification, whose rate are pretty high in some cases, can be pointed out.

Besides validation data reference, *CAdES-X Long* brings validation data itself which makes it possible to verify a signature without performing external queries if it is acknowledged that CAs' keys cannot be compromised. Whereas this approach offers a faster signature verification, its main disadvantage is a greater final size of signed document, since the validation data may even demand more storage space than the signed client's content. Furthermore, this approach may cause data replication when different documents signed by the same signer keep their own copy of validation data, e.g. the signer's certificate.

In both cases, when CAs' keys compromise is a possibility, an additional time-stamp must be included over the validation data as a means to prove that it has been gathered in time, i.e. before an attacker is able to forge these information. The signature in question (*CAdES-X-Long Type 1 or 2*), however, will be accepted as valid until any certificate from the certification path of Time-Stamp Authority (TSA), which issues that time-stamp, expires or is revoked.

Finally, for those signatures supposed to be valid even after TSA's certificate expiration or revocation, a new time-stamp must be included, which originates the *CAdES-A* format, as illustrated in Figure 5. This time-stamp is known as an archive time-stamp and it embraces all the signature elements, including the document, the signed and unsigned attributes as well as the signature itself. However, whenever the archive time-stamp is near to becoming invalid due to its cryptographic algorithm weakness or limits in the validity period of the TSA certificate, a new one must be included.

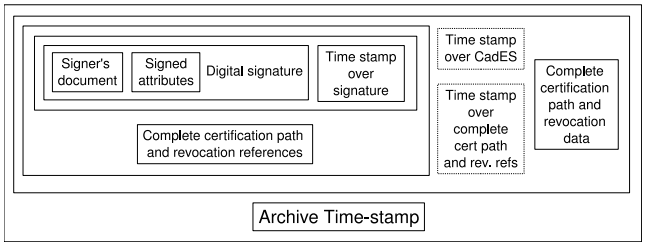

Figure 5: *CAdES-A* format.

2.3 Example

This subsection presents a real example to illustrate the validation of signed documents. We have chosen an electronic document in *CMS* format that was signed using a digital certificate from *ICP-Brasil* [16], the Brazilian national PKI, as illustrated in Figure 6. First, there is a signer's certification path which is composed of four certificates: *User*, *Serasa SRF CA*, *SRF CA* and *Brazilian Root CA*. Also, CAs issue CRLs that are named as CRL_1, CRL_2, CRL_3.

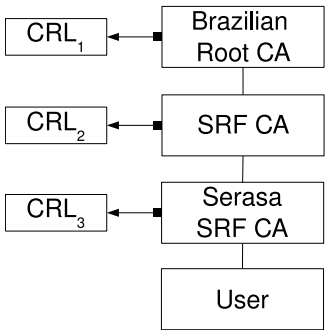

Figure 6: ICP-Brasil user's certification path.

First, validation data and a signature using a 2048 bits RSA key were measured and their sizes in DER codification are shown in Table 1. Note that CRL_3 has a significant size, since it is a large Certificate Revocation List containing 7442 revoked certificates, which was issued on 14 August 2008. It is important to mention that long CRLs such as CRL_3 are often present in PKIs, mainly those which remain revoked but expired certificates as a means to support users to verify a signature created before the signer's certificate expired.

Table 1: Validation data's and signature's size.

Element	**Size** (Bytes)
Signature	256
User	1076
Serasa SRF CA	1301
SRF CAs	1109
Brazil Root CA	1212
CRL_1	534
CRL_2	487
CRL_3	201,390
Total	207,109

In the next step, the creation of a *CMS* whose encapsulated content is the Serasa SRF CA's CPS, a PDF document of 263,503 bytes, was simulated. Then, that CPS is signed by a 2048 bits RSA key and the whole signer's certification path and related CRLs are included into the *CMS* archive.

As a consequence, that *CMS* file measured 471,098 bytes. Therefore, note that only 56% of total size (263,503 of 471,098 bytes) corresponds to the PDF document which highlights a great amount of data spent for the sake of signature verification. Additionally, it is important to point out that the signed content is not a small document and the signature was not time-stamped which means the inclusion of another signed document. In this case, the percentage of signed content would decrease even more.

Finally, from this example one can comprehend the high cost of validation data on the size of the signed document. Then, users will choose the best signed document format in accordance with their domain requirements and their expectations of the signature verification process efficiency.

3. RELATED WORK

The main drawback in the use of certificates to bind signer's ID of an electronic document to his public key is the requirement of including into the data structure of the signed document evidence that the certificates were valid when the document was signed. This makes the document large and inefficient to verify due to the great number of cryptographic operations that are performed each time its signature is checked. Moreover, to keep the signature time anchored, it is periodically necessary to add time-stamps to the documents. This makes the size of the document increase over the time.

From the signer's point of view, the ideal scenario would be using a self signed digital certificate to sign electronic documents. In this scenario, each recipient of the electronic document should trust only in the signer's certificate. From the recipient's point of view, the ideal would be only one trusted root certificate issuing all signer's certificates. And, each one with a validity period as small as possible. Both scenarios are unpractical and unreal.

Thus, more complicated certificate arrangements are used to deploy a PKI. It is usual to find in each user application a set of root certificates that the user trusts. And, subordinate to each root certificate, many certification authorities are created to be responsible for policies and issue certificates to final users. To securely manage these structures many components are necessary, like time-stamp, registration and certification authorities. Each one of these components issues information that is included in the electronic documents to make them verifiable.

To overcome the problems that arise in the use of this more complicated scheme, different strategies in the literature were proposed. One of the most know is the short-term certificates.

Short-term certificates are certificates issued to be valid for a short period of time, making it unnecessary to revoke [17]. Thus, the data structure can be small since is not necessary to include certificate revocation lists (CRLs) in the electronic document. However, even though the signer's certificate is not required to verify its validity against a CRL, all other certificates of the path between the signer's certificate and a root certificate maintain earlier revocation scheme requirement.

Nested certificates, introduced by Levi[18], is another proposal to optimize the number of cryptographic operations to verify a document signature. They are new certificates that are issued for the certificates, resulting in a nested certificate path whose validation is performed by attesting the signature of the first nested certificate in the chain, and then easily verifying others by computing and comparing only hashes.

All known proposals have advantages and disadvantages and, until now, there are no large scale implementations using them. Besides theses proposals, an efficient scheme to sign and validade signed electronic document using digital certificates is still an open problem.

4. OPTIMIZED PKI

In this section, the Optimized PKI (OPKI) architecture is presented. The main target of this architecture is twofold: a) to minimize the signature size in existing documents; b) to reduce the effort for certificate path verification. The main assumption of this work is that more efficient revocation techniques are an adequate methodology to improve signed electronic documents. It is worth mentioning that this process is an expensive task and it often involves a large amount of data, as CRLs increase constantly.

The OPKI has been proposed to be a conventional PKI that also issues a new kind of X509 certificate: the Optimized Certificates (OC). It is based on a well known solution to avoid revocation schemes, called short-term certificate [17]. This original proposal defines a short period for the certificate validity, for example, a day. As a consequence, the probability that a revocation occurs is very low. In this work, we are proposing to set the certificate validity only to a particular time instant k, i.e., the *validity* field of a X509v3 begins at k and ends at k. In other words, the validity refers to the moment when the OC was issued. Therefore, there is no reason to revoke an OC, since if it was issued, it was valid.

The OC has been proposed to replace the conventional signer's certificate and additional validation data in a signed document, in order to make up a certification path that is

verified more quickly. However, an OC should not optimize an invalid certification path, therefore the signer's certificate is verified. Once it is attested, all its information useful to identify the signer and to verify his signature, such as the distinguished name and public key, are embedded in the OC, which is issued. Therefore, the OC issuance is a kind of proof of a document's signature validity which may also be attested using the signer's information embedded in OC.

Additionally, another remarkable advantage of OC is that it could eliminate time-stamps, as the OC validity is specified for a particular time instant. It is achieved by including the signed document's hashcode into OC. Furthermore, an OC whose signature algorithm is going to phase out soon may be replaced by another OC with a stronger hash algorithm. Thus, it is possible to make a document's signature lifetime longer without raising the amount of validation data.

The OPKI architecture proposes a simple and effective solution to implement the OC issues, it is composed of an Optimized Certificate Certification Authority (OCCA) and a Crypto Time (CT) service (Figure 7). This Figure illustrates an OPKI architecture that operates together with a traditional PKI, it can be observed that there are traditional (U_1, U_2, U_3) and optimized certificates (OC_1, OC_2). Numbers ranging from 1 to 6 highlight the relationship between certificates and its revocation status source. Number 5 shows OCCA acquiring Novomodo proofs from Crypto Time in order to verify its revocation status. These proofs have been made available by the Root CA. However, note that 6 shows that Root CA may retrieve it revocation status from Crypto Time or CRL_1. Number 7 points out the relationship of traditional end user's certificates, which are valid for a period, and OCs that are valid for a specific time instant. Finally, there are three values inside of an OCCA certificate. Such values are related to Novomodo parameters - *validity target* (X_n),*revocation target* (Y_1) and *granularity* (l). More details about this architecture are given below.

The OCCA is an online service specially developed for users to submit all signed document's information, such as hashcode, signature and its time-stamp, signer's certificate path, as well as the respective CRLs or OCSP responses. The OCCA's duties are used to validate the signature and return an OC to the requester. Considering that a long certificate path verification demands higher resources for the validation task, the distance between an OC and a Root CA must be as short as possible. Thus, the OCCA might be placed on the top of a PKI, being a Root CA. However, online Root CA strategy to issue certificates for end users is not a secure choice. Therefore, aiming at both PKI security and shortest certificate path validation, an OCCA directly subordinated to a Root CA, which can be kept offline, is proposed.

Another important aspect of the OPKI architecture is that both OCCA and Root CA revocation status are required. Then, aiming at an efficient revocation solution, the use of the Novomodo method that deploys small and fixed-size revocation status proofs [19] is proposed. This method, that is described in details in the next subsection, quickly determines whether OCCA's certificate is valid through hash function evaluations. The Novomodo revocation status data is produced by an offline "Root CA", safely stored in an online entity named here as Crypto Time and later used by the OCCA.

Finally, it is important to mention that the OC keeps the compatibility with X509v3 standard, because there is no field modification nor new information inclusion, as Novomodo parameters and time-stamp characteristics are implemented through X509v3 extensions.

4.1 OPKI Revocation Scheme

Requiring an efficient revocation scheme, OPKI employs Novomodo. This method was proposed by Micali as an attractive alternative for CRL or OCSP when dealing with signed documents. The main advantages of this method are related to validity and revocation proofs, which can be easily produced by hash functions, being their output size fixed and significantly smaller than a CRL or an OCSP response. Thus, these proofs are easily fetched and stored by OPKI users.

Additionally, status verification in Novomodo method consists of evaluating hash functions and comparing short and fixed-size hashcodes. These operations are significantly faster than extracting revocation status from CRLs and OCSP responses whose content is signed and therefore a signature must be attested.

Based on the original idea of the Novomodo method and the OPKI scenario (Figure 7), it works as follows: the Root CA chooses a hash function F, the OCCA's certificate lifetime t, a granularity l and two random values X_0 and Y_0, keeping these last two private. Finally, l is the time interval between two consecutive revocations and the number of time intervals is given by $n = t/l$.

The Root CA applies n times the value X_0 to the function F resulting in the value X_n, called *validity target*. Y_0 is applied once to F, resulting in Y_1, named *revocation target*. Afterwards, the Root CA issues OCCA's certificate including both targets and the granularity encoded as a X509v3 extension named *NovomodoTargets* (Section 4.5). Later, for each k ranging from 1 to $n-1$, Root CA published the *validity proof* (X_{n-k}), evaluating $F^{n-k}(X_0)$, whether OCCA's certificate is still valid, otherwise the *revocation proof* (Y_0). Later, as a means to attest the OCCA's validity both OCCA and users just need to acquire the last proof from Crypto Time as well as the Novomodo parameters (X_n, Y_1 and l) from OCCA's certificate and assure that $F^k(proof)$ equals X_n. If the comparison fails, it means that OCCA status is revoked which can be assured by comparing $F(proof)$ and Y_1.

It is important to point out that due to security reasons Root CAs are often kept offline and they are turned on just for issuing certificates or CRLs. Moreover, the Root CA of an OPKI is also responsible for publishing Novomodo proofs, which are supposed to be issued at the same time and in the same frequency as CRLs for the sake of the Root CA being turned on as little as possible. However, that premise can not be followed as OPKI policy may require Novomodo proofs to be issued in a higher rate, e.g. daily.

Therefore, in order to overcome that drawback, the use of an online entity called Crypto Time is proposed. It incorporates a slight modification on Novomodo's method, in which before the Root CA goes offline, all the proofs will be produced and securely stored into the Crypto Time instead of Root CA going online to publish each of $n-1$ proofs. The Crypto Time is responsible for keeping secret the k^{th} validity proof until OCCA's auditors have assured OCCA's integrity on the k^{th} day and consecutively allowing Crypto

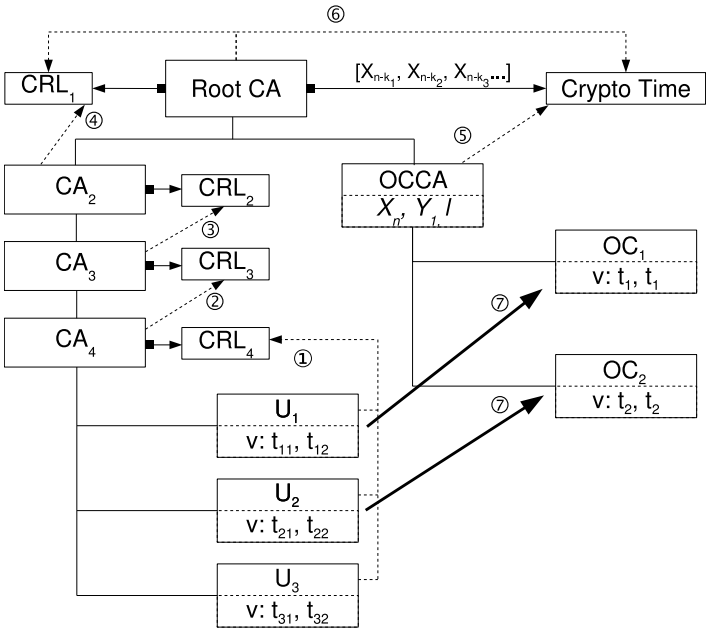

Figure 7: Optimized PKI.

Time to reveal X_{n-k}. Once OCCA's private key's compromise is detected by auditors, they ask Crypto Time to publish Y_0 which revokes OCCA's certificate - note that to produce a fake validity proof X_{n-k} from the previous one ($X_{n-(k-1)}$) is unfeasible since $X_{n-k} = F^{-1}(X_{n-(k-1)})$.

Figure 7 also shows that both CA_2 and $RootCA$ check CRL_1 in order to verify their validity. Since $RootCA$ is responsible for revoking itself and CA_2, we can figure out the same scheme in Novomodo's method: a Novomodo proof can assure validity for both $OCCA's$ and $RootCA's$ certificates. Therefore, it is possible to eliminate all CRLs in the OC's certification path.

4.2 Time of Signature Validation

The main characteristic of the OC is the absence of revocation, as its validity corresponds to a specific time instant k: the moment the OCCA issues the OC. However, an OC attests the signature of a document and the respective signer's certificate, therefore the time reference used to validate that certificate prior the OC issuance should be embedded in the OC. Then, this information is encoded as a X509v3 extension named *validationTime* that has been defined in Section 4.5.

There are five alternatives for the time reference used to attest the certificate of the document's signer and they are related to the time sources listed in Table 2. In the first case, the signer's certificate is verified at the particular time when an OC is requested. This corresponds to $N = 1$ in Table 2. In the second case, the signer's certificate is verified at the time encapsulated by a time-stamp ($N = 2$). In the third case, a signature has already been optimized in the past but it is submitted again for the sake of replacing its OC. This refers to the time-source $N = 3$. In the fourth case ($N = 4$), the time reference means the signing time and it should only be provided by the document's signer who proves key ownership through challenge-reply protocol. Finally, $N = 5$ is given by a verifier who holds a signed document and wants to improve the signature.

Table 2: Time reference for signature validation.

N	Reference	Source	Trust
1	OC request instant	The OCCA's internal clock.	Trusted
2	Past	A time-stamp of document's signature.	Trusted
3	Past	Another previous OC.	Trusted
4	Past	The document's signer.	Untrusted
5	Past	A document's verifier.	Untrusted

Finally, it is important to mention the trust in these five alternatives. Besides time sources $N = 4$ and $N = 5$, others can be trusted. The first one ($N = 1$) takes the internal time from the clock of an OCCA, which is supposed to employ the same synchronization and security measures that a trusted TSA maintains. The following two time sources, $N = 2$ and $N = 3$, are based on artefacts emitted by trusted entities. Time source $N = 4$ corresponds to a signing time claimed by the signer. In general, however, verifiers do not relie on it, except for particular contexts such as the notary one. The last one ($N = 5$) is not accepted as trusted because an attacker, who is aware of the revocation date of the signer's certificate, may ask OCCA to validate a forged signature at an earlier time when the signer's certificate was valid.

4.3 OC's Relative Time-stamp

Although the OCCA is supposed to work in a proper way, measures should be taken to guarantee or at least to reduce the chances of a malicious OCCA issuing fake OCs. Therefore, a relative time-stamp [20] scheme is proposed as a means to support auditors to track OCCA's operations. Fur-

thermore, OCs carry relative time-stamp information, which is encoded as a X509v3 extension named *relativeTimeStamp* that has been defined in Section 4.5.

The relative time-stamp scheme is deployed in order to chain all OCs issued by an OCCA. Each OCCA has its own chain of hashcodes whose initial value is the Novomodo validity proof X_{n-k} with $k = 0$, i.e. the first validity proof in the OCCA's lifetime that is also the validity target (X_n). Each OC and validity proof is included in the chain in the same order they are issued by OCCA and Root CA, respectively. Finally, this process ends when the OCCA's certificate expires.

Figure 8 presents the chaining of OCs issued throughout three periods of l length, where l is the granularity of Novomodo method. The first interval refers to the time $k = 0$, thus the chain is initiated with the value X_n. Then OC_1 is issued and bound to X_n, subsequently OC_2 is issued and also linked to OC_1's hashcode computed with F function. As soon as the second period ($k = 1$) begins, hashcode of the concatenation of the new validity proof (X_{n-1}) with the hashcode of the last OC from the previous period (OC_x) is computed and then it is pushed in the chain. Thus, the same linkage process is repeated for $k = 2$ and $k = 3$. Note that in $k = 4$ period no OC can be issued anymore, since the OCCA's certificate has expired and the lack of proof blocks OCs' issuance. Finally, the chain is marked as closed through the inclusion of the secret value X_0, which may have been disclosed by Root CA or Crypto Time.

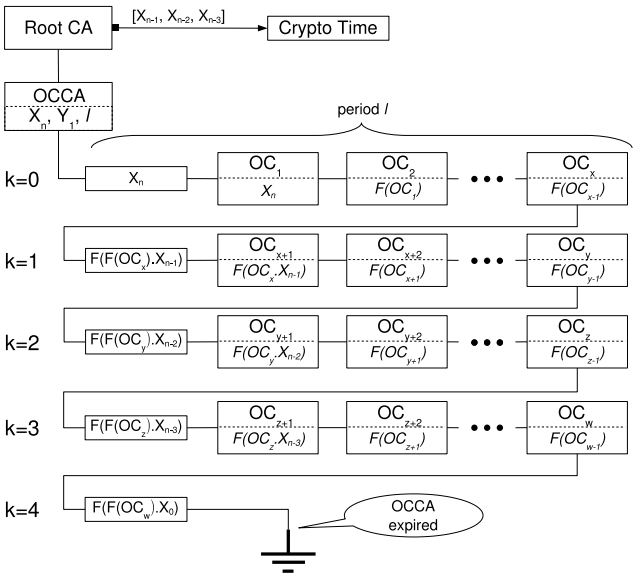

Figure 8: OC linking.

Novomodo validity proofs in the chain are useful to give further information to the relative time-stamp scheme, thus it is possible to know in which of n Novomodo periods an OC was issued. Also, those proofs work as checkpoints that guarantee OCCA's integrity until that time, which makes it possible to identify which OCs were issued before OCCA's compromise, for example.

4.4 Long Term Archiving Benefits

The benefits of Optimized Certificate employment are even more significant for signatures that must be verifiable for a long period. Those signatures are generally preserved using the successive time-stamping approach, such as those employed in CAdES-A. This approach, also used in XML Advanced Electronic Signature (XAdES) and Evidence Record Syntax (ERS) [21], is the most widely used method to achieve long term preservation of digital signatures, and it has been subject of normative initiatives in several countries, including Brazil.

However, this method has costs that are unacceptable in many scenarios, especially, resource-constrained environments. The ever-growing storage and processing requirements for these signatures validation, although, could be reduced by replacing all signature validation data, i.e. certificates, certificate revocation data, and time-stamps; with an OC and its certification path.

As time goes by, however, the OC itself, as an archive time-stamp, may become invalid due to its cryptographic algorithm weakness or limits in the validity period of the OCCA certificate. In this case, if the signature still needs to be validated, the current OC and its certification path must be replaced by new ones. This task could be done by sending the old one to the OCCA, and receiving a new OC.

By doing so, differing from the successive time-stamping approach, we get relatively fixed requirements and a lightweight method for verifying digital signatures on a long term basis.

4.5 X509v3 Format and OC's Extensions

One of the main concerns of this work is to preserve the X509 certificates' standard for the purpose of making OC interoperable with existing X509 based applications. Therefore, besides OC's validity which is carried in X509 *validity* field, all other information is embedded in OC using X509v3 extensions.

Figure 9 illustrates the X509 ASN.1 syntax that defines the Optimized Certificate, whose fields structure is described afterwards.

```
OptimizedCertificate  ::=  SEQUENCE  {
  version         [0]  EXPLICIT Version DEFAULT v1,
  serialNumber         CertificateSerialNumber,
  signature            AlgorithmIdentifier,
  issuer               Name,
  validity             Validity,
  subject              Name,
  subjectPublicKeyInfo SubjectPublicKeyInfo,
  issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                       -- If present, version MUST be v2 or v3
  subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                       -- If present, version MUST be v2 or v3
  extensions      [3]  EXPLICIT Extensions OPTIONAL
                       -- If present, version MUST be v3
}
```

Figure 9: ASN.1 syntax of OC.

version version 3.

serialNumber unique identifier for OC.

signatureAlgorithm indicates the algorithm used to calculate the digital signature.

issuer OCCA Distinguished Name (DN).

validity $t_1 = t_2 = t$. It means the time of OC's issuance.

```
id-ce-novomodo-validity-proof OBJECT IDENTIFIER
::= { id-ce not defined yet }
Novomodo-proof ::=  OCTET STRING
```

Figure 12: ASN.1 syntax of extension novomodo-validity-proof.

subject copied from the traditional certificate of the document's signer.

subjectPublicKeyInfo copied from the traditional certificate of the document's signer.

extensions a sequence of extensions which carry information related to the digest of the signed document (Figure 10), the time of document's signature validation (Figure 11). and relative time-stamp information (Figure 13). These extensions are respectively defined bellow using ASN.1 syntax.

```
id-ce-signedDocDigest OBJECT IDENTIFIER
::= { id-ce not defined yet}
SignedDocDigest ::= SEQUENCE {
    digestAlgorithm     OBJECT IDENTIFIER,
    docHash             OCTET STRING
}
```

Figure 10: ASN.1 syntax of extension *SignedDocDigest.*

```
id-ce-validationTime OBJECT IDENTIFIER
    ::= { id-ce not defined yet }
ValidationTime ::= SEQUENCE {
 timeSource  INTEGER,
 time        TIME }
```

Figure 11: ASN.1 syntax of extension *ValidationTime.*

```
id-ce-relativeTimeStamp OBJECT IDENTIFIER
::= { id-ce not defined yet }
RelativeTimeStamp ::= SEQUENCE {
    algorithmId     OBJECT IDENTIFIER,
    previous        OCTET STRING }
```

Figure 13: ASN.1 syntax of extension *RelativeTimeStamp.*

Note that in order to deploy the proposed Novomodo method it is needed to publish *validity target*, *revocation target* and *granularity*. Therefore, those information are embedded in OCCA's certificate as a X509v3 extension whose ASN.1 syntax in Figure 14

```
id-ce-novomodoTargets OBJECT IDENTIFIER
    ::= { id-ce not defined yet }
NovomodoTargets    ::=    SEQUENCE {
    algorithmId              OBJECT IDENTIFIER,
    validityTarget           OCTET STRING,
    revocationTarget         OCTET STRING,
    granularity              INTEGER }
```

Figure 14: ASN.1 syntax of extension *NovomodoTargets.*

5. OPKI DEPLOYMENT

In practice an Optimized PKI can be seen as a way of reducing digital signature storage requirements and its validation processing as well as bandwidth costs within a given domain.

For instance, Figure 15 illustrates two domains in the same PKI. Domains are distinguished by their participants' color. Users, which are placed on the botton of PKI, can optimize their signatures within their domain through OCs. Also, each domain has its particular OCCA for the purpose of attending its users' requests.

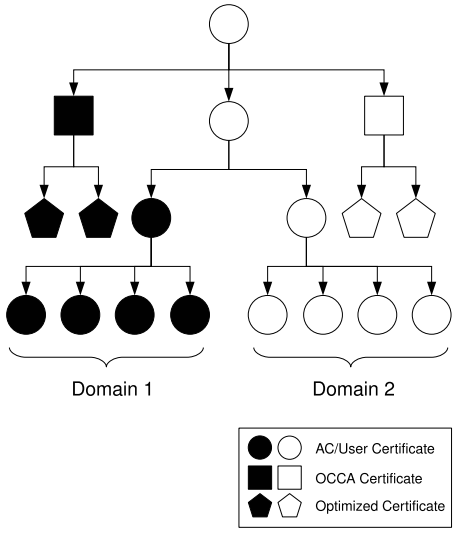

Figure 15: Two domains in a PKI.

Moreover, a signed document entering a domain may have its validation data replaced by an Optimized Certificate and its certification path. If the OCCA does not belong to the domain, this procedure may be followed by an one-time digital signature validation with the original validation data in order to attest OC's accuracy.

Once an OC is recognized as valid within the domain, either because of the OCCA is trusted, or because its response accuracy were evaluated, subsequent validations of the signature are optimized by the OC's employment. This optimization is relative to the minimized storage requirements for the digital signature, the reduced processing in its validation, and the minimal bandwidth costs in gathering revocation information as well as in transferring the new validation data to others within the domain.

In the long term however, an archive may be needed to preserve the digital signature with the original validation data in order to maintain its validity in case of it being re-

quired outside of this domain. Moreover, preserving original validation data allows a user who contests an OC accuracy to validate the original signer's certification path at the time the OC attests a document's signature. Thus, users of a domain choose to trust or not in an OCCA since they are able to check OCs.

Figure 16 illustrates a domain in which users do not trust in OCCA, consequently they only consider an OC to be valid if the traditional signer's certificate related to that OC can be verified. Thus, numbers 1 and 6 show signed documents coming into a domain. Number 6 brings an OC that is rejected since the traditional signer's certificate is not available for validation. Number 1 brings the whole traditional validation data and it is submitted to an OCCA and the domain's archive which has long term archiving duties. Once OCCA attests the document's signature an OC is issued and attached to the signed document replacing the traditional validation data (number 2). Afterwards, users receive the signed document (number 3) but they need to check the OC's accuracy. Then, users acquire from the domain's archive the traditional validation data (number 4) and validate the document's signature. Once it is validated, users accept OC. Later, whenever users deploy the signed document, they verify the signture using the optimized validation data with Novomodo proofs from Crypto Time (number 5).

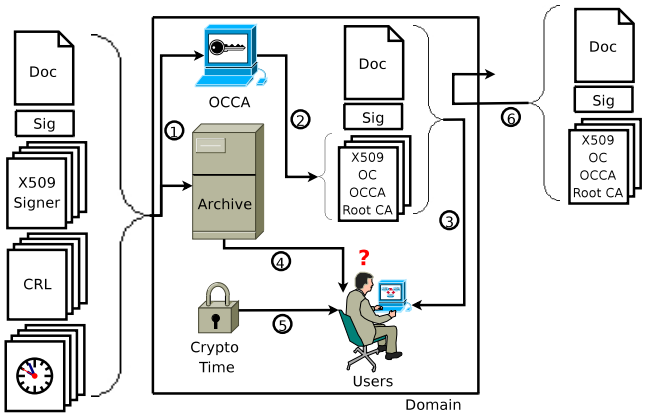

Figure 16: Untrusted OCCA in a domain.

Futhermore, a domain may offer an OCCA in which users can trust and use it as an online service (Figure 17). This scenario is quite similar to the previous one (Figure 16), however, users do not need to verify the document's signature as shown in number 4 before accepting an OC. Moreover, a user, who desires to send a signed document to another outside the current domain, should replace the optimized validation data as shown in Figure 17. Number 1 highlights users sending a signed document in which is embedded an OC. Number 2 and 3 shows the replacement of optimized validation data by the traditional one gathered from domain's archive.

6. SIGNATURE VALIDATION COST

An OPKI compares favorably to a traditional PKI in analysis of performance validation processes, as the number of signature validations during the lifetime of an electronic document is greatly reduced.

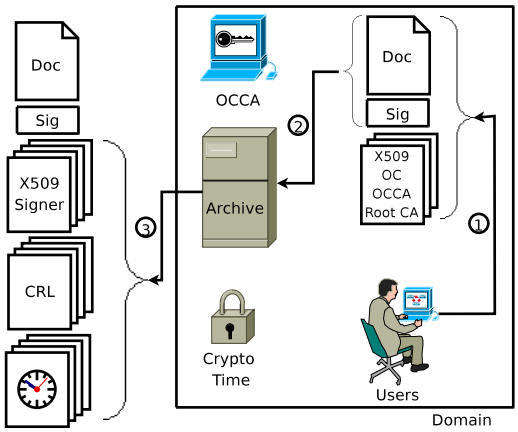

Figure 17: A signed document leaving a domain with a trusted OCCA.

In contrast to the signature generation process which occurs only once, the signature verification is normally repeated innumerable times. This validation can be made by different entities at different times. Thus, in evaluation terms, the cost functions of a conventional signature and signatures using OCs are in terms of the computational effort demanded by verifiers and signers.

Most of the processing power in traditional PKI deployments is used to validate digital certificates. With OPKIs, only two additional certificates need to be verified, the OCCA certificate and the OC. The OCCA certificate is verified by the user through the hash function of Novomodo, by $F^k(X_{n-k})$, where k is a specific instant of OCCA's lifetime.

The values of the Novomodo targets (X_n and Y_1) are contained in the OCCA certificate and the proofs (X_{n-k} or Y_0) are obtained from Root CA's publications or Crypto Time, as discussed in Section 4.1.

After simple calculations the revocation status can be confirmed. This is sufficient to verify the signature without consulting any additional information. The OCs do not need to be verified for revocation since it is only validy for an instant and then it could not be revoked.

6.1 Complexity in the classical scheme

The signatures of electronic documents consist of a data set, which allows the verification of the integrity and authenticity of an electronic document. These are the following sequences of bits; the signer's private key Sig, the signature's time-stamp TS; the chain of digital certificates CC, and the list of CRLs RR, each list associated to a particular certificate.

Signature validation includes verifying if the subscribers' digital certificates and the entire certification chain were valid on the date contained in the time-stamp, and also the validation of the time-stamp itself.

Using a traditional PKI, in the lifetime of an electronic document, the total number of digital signatures used for verification is given as $S = 2m(n_1 + n_2 + 1)$, where $n_1 \geq 2$ is the number of certificates in the CC, $n_2 \geq 2$ the number of certificates in the chain used to verify the time-stamp TS, and $m \geq 1$ is the number of times that the signature of the document is verified. According to this equation, using the conventional method, the complete validation of one signa-

ture demands the checking of the document signature and all the respective certificates and CRLs of the certification chain each time the document needs to be verified.

6.2 Complexity in the optimized scheme

The total number of digital signatures that should be performed to verify an electronic document - in an OPKI context - is given by $S = 4m$ where m is the same as stated in Section 6.1. This takes into account that four signature verifications are necessary: the document signature; the OC signature; the OCCA's certificate signature and the Root CA certificate signature.

However, considering the OCCA's certificate publication and confidence in the Root CA, it may also not be necessary to verify the Root CA certificate. This means that less computational power is required. In this way, the total number of digital signatures needed for verification is reduced to $S = 3m$, which is significantly lower than the classical scheme.

The amount of memory used for storage of the data structure of a electronic document's digital signature in a classical PKI is given by $E = 2(n_1 + n_2)$. In comparison, an OPKI requires only 4, which means a reduction of at least 50%.

7. CONSIDERATIONS

The signer must validate his digital certificate when he issues the signatures of electronic documents. And, the recipients must validate the signatures each time they need to verify the documents' authenticity. These tasks require many cryptographic signature validations. A reasonable strategy to avoid an overload on the recipients' computers would be to delegate those computations to an external service.

The delegation of the digital certificates validation to a third party entity is not a new idea. Peifang Zheng [22] calls this entity a verifier. The protocol Online Certificate Status Protocol (OCSP) [10] is considered a verifier. OCSP is an Internet protocol used for obtaining the revocation status of a digital certificate. It was created as an alternative to certificate revocation lists (CRL), specifically addressing certain problems associated with using CRLs. However, this protocol cannot be used as evidence that a certificate was valid in the past without additional services like time-stamping. The verifier could be used either by the signer or by the recipients to improve the speediness of certificate and signature validation. Although this strategy may appear attractive, external queries are needed as a means to consult verifiers, whose responses require signature validation to guarantee authenticity and integrity.

The OCCA is similar to the conventional verifier and the OC is compared to the conventional verifier's response. However, the OC can work as an evidence that a document's signature and the respective signer's certificate were validated in the past without a time-stamp. This is because the time and the accuracy of that validation have already been attested by the OCCA at the time it signes the OC. Also, the OC's signature is easier to verify than the conventional verifier response's due to the OC's short certification path and fast revocation scheme.

The fact that the OC is valid for only one k instant gives the OC the same positive features described by Rivest [17], where a certificate has low probability of revocation soon after its emission. Moreover, the method does not present the problems of short-term certificates [23, 24, 25] as a generation of new cryptographic keys and certificates each time that the subscriber needs to renew its certificate.

A positive point of the OPKI approach is that even having the OCCA's private key at any k instant, an attacker can only issue false OCs until k ends since he can not forge a fake proof of OCCA's certificate validity in $k + 1$. Therefore, to compromise the OPKI an attacker needs to compromise both the OCCA's private key and the source of Novomodo proofs: the Crypto Time or the Root CA.

In the case of any fraud that should occur, the attacker may use a compromised Root CA's private key to create a false OCCA. However this problem is easily countered by publication of the hash of the true OCCA certificate in a public directory.

Users should register the trusted OCCAs by taking hashes from the public directory and saving them into their computers before beginning to use the OCCAs.

Besides the OC optimizes signed document's verification, it also improves signature format that has been discussed in Section 2. The proposed approach requires a short certification path for an OC as well as dismisses CRLs. Moreover, the OC replaces time-stamps, which also requires verification data like the Time-Stamp Authority's certification path and related CRLs.

8. CONCLUSION

This paper has described the Optimized PKI in which signed documents can take advantage of the Optimized Certificate: a new kind of X509 certificate that improves both the signature's format and validation. These improvements are achieved because OCs are based on short-term certificates, short certification paths and efficient revocation schemes, therefore signed documents are easily verified and require a lightweight validation data. Thus, OPKI is an attractive solution to domains where signed contents could not be used due to resources restrictions.

The next step in this work is to design a revocation solution in which the revocation status of Root CA and OCCAs can be attested without external query. Once this scheme is designed, the OC will embedded revocation status of its certification path and then signed document will be self-verifiable. Furthermore, OCCA and OC based applications will be implemented for the sake of evaluating OPKI in practice.

9. REFERENCES

[1] L. M. Kohnfelder, Towards a practical public-key cryptosystem, Bachelor's thesis, MIT (1978).

[2] C. Satizábal, J. Hernández-Serrano, J. Forné, J. Pegueroles, Building a virtual hierarchy to simplify certification path discovery in mobile ad-hoc networks, Comput. Commun. 30 (7) (2007) 1498–1512.

[3] D. Dzung, M. Naedele, T. Von Hoff, M. Crevatin, Security for industrial communication systems, Proceedings of the IEEE 93 (6) (2005) 1152–1177.

[4] A. Willig, Recent and emerging topics in wireless industrial communications: A selection, IEEE Transactions on Industrial Informatics 4 (2) (2008) 102–124.

[5] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation

List (CRL) Profile, RFC 5280 (Proposed Standard) (May 2008).
URL http://www.ietf.org/rfc/rfc5280.txt

[6] C. Adams, P. Cain, D. Pinkas, R. Zuccherato, Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP), RFC 3161 (Proposed Standard) (Aug. 2001).
URL http://www.ietf.org/rfc/rfc3161.txt

[7] R. F. Custódio, M. A. G. Vigil, J. Romani, F. C. Pereira, J. da Silva Fraga, Optimized certificates - a new proposal for efficient electronic document signature validation., in: S. F. Mjolsnes, S. Mauw, S. K. Katsikas (Eds.), EuroPKI, Vol. 5057 of Lecture Notes in Computer Science, Springer, 2008, pp. 49–59.

[8] S. Lloyd, Understanding certification path construction, PKI Forum (2002) 1–14.

[9] R. Sparks, Actions Addressing Identified Issues with the Session Initiation Protocol's (SIP) Non-INVITE Transaction, RFC 4320 (Proposed Standard) (Jan. 2006).
URL http://www.ietf.org/rfc/rfc4320.txt

[10] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP, RFC 2560 (Proposed Standard) (Jun. 1999).
URL http://www.ietf.org/rfc/rfc2560.txt

[11] B. Kaliski, PKCS #7: Cryptographic Message Syntax Version 1.5, RFC 2315 (Informational) (Mar. 1998).
URL http://www.ietf.org/rfc/rfc2315.txt

[12] R. Housley, Cryptographic Message Syntax (CMS), RFC 3369 (Proposed Standard), obsoleted by RFC 3852 (Aug. 2002).
URL http://www.ietf.org/rfc/rfc3369.txt

[13] E. T. C. E. Signatures, I. (ESI), Electronic signatures and infrastructures (esi); cms advanced electronic signatures (cades), Tech. rep., European Telecommunications Standards Institute (oct 2008).
URL http://webapp.etsi.org/workprogram/Report_WorkItem.asp?WKI_ID=28069

[14] D. Pinkas, N. Pope, J. Ross, CMS Advanced Electronic Signatures (CAdES), RFC 5126 (Informational) (Mar. 2008).
URL http://www.ietf.org/rfc/rfc5126.txt

[15] E. T. C. E. Signatures, I. (ESI), Electronic signatures and infrastructures (esi); profiles of xml advanced electronic signatures based on ts 101 903 (xades), Tech. rep., European Telecommunications Standards Institute (nov 2008).
URL http://webapp.etsi.org/workprogram/Report_WorkItem.asp?WKI_ID=22942

[16] Instituto Nacional de Tecnlogia de Informação, Portal ICP-Brasil (oct 2008).
URL https://www.icpbrasil.gov.br/

[17] R. L. Rivest, Can we eliminate certificate revocations lists?, in: FC '98: Proceedings of the Second International Conference on Financial Cryptography, Springer-Verlag, London, UK, 1998, pp. 178–183.

[18] A. Levi, M. U. Caglayan, C. K. Koc, Use of nested certificates for efficient, dynamic, and trust preserving public key infrastructure, ACM Trans. Inf. Syst. Secur. 7 (1) (2004) 21–59.

[19] S. Micali, NOVOMODO: Scalable Certificate Validation and Simplified PKI Management, in: Proceedings of the 1st Annual PKI Research Workshop, NIST, Gaithersburg MD, USA, 2002.

[20] S. Haber, W. S. Stornetta, How to time-stamp a digital document, in: CRYPTO '90: Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology, Springer-Verlag, London, UK, 1991, pp. 437–455.

[21] T. Gondrom, R. Brandner, U. Pordesch, Evidence Record Syntax (ERS), RFC 4998 (Proposed Standard) (Aug. 2007).
URL http://www.ietf.org/rfc/rfc4998.txt

[22] P. Zheng, Tradeoffs in certificate revocation schemes, SIGCOMM Comput. Commun. Rev. 33 (2) (2003) 103–112.

[23] P. McDaniel, A. D. Rubin, A response to "can we eliminate certificate revocation lists?", in: FC '00: Proceedings of the 4th International Conference on Financial Cryptography, Springer-Verlag, London, UK, 2001, pp. 245–258.

[24] C. A. Gunter, T. Jim, Generalized certificate revocation, in: POPL '00: Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, ACM, New York, NY, USA, 2000, pp. 316–329.

[25] V. Ungureanu, Formal support for certificate management policies, Computers & Security 23 (4) (2004) 300–311.