Cleaning up the PKI for Long-Term Signatures

Martín A. G. Vigil[1], **Ricardo Felipe Custódio**[2]

[1] Kryptographie und Computeralgebra
Technische Universität Darmstadt
Hochschulstraße 10
64289 Darmstadt, Germany

[2]Laboratório de Segurança em Computação (LabSEC)
Universidade Federal de Santa Catarina (UFSC)
PO Box 476
88040-900 Florianópolis, Brazil

vigil@cdc.informatik.tu-darmstadt.de, custodio@inf.ufsc.br

Abstract. *In this paper we present a new approach for the conventional X.509 Public Key Infrastructures (PKI). Our goal is to reduce the effort to handle signatures in the long term. The novelty is that a Root CA reissues subordinate certificates of final users, but adjusting validity periods to exclude the periods after a revocation. The Root CA also authenticates timestamps. The result is the cleaned PKI, which is simpler than the conventional PKI because: a) there is no revocation; b) there is no intermediary Certification Authority; c) signatures are trustworthy as long as the used cryptographic algorithms remain secure. As benefits, we reduce the need of timestamps and consequently the demand for storage space and processing time to use signed documents.*

1. Introduction

Digital certificates based on the X.509 Public Key Infrastructure (PKI) and digital signatures are the most prominent means to ensure integrity, authenticity and proof-of-existence for digital documents. However, this solution presents several issues (see [Gutmann 2002] for an overview). One of the hardly criticized drawbacks of PKI is revocation. For instance, a certificate can be revoked *before* it expires, e.g. because of key compromise. But, what if we could foresee when a certificate would be revoked? Then, we could set certificates in advance to expire *before* they are revoked. By doing so, we eliminate the need for revocation, thereby *simplifying* PKI. Base on this idea, we present our proposal.

Contribution We present a new proposal for simplifying the conventional X.509 PKI by removing revocation and intermediary Certification Authorities (CA). Our idea consists of reissuing all final certificates in a PKI, but assigning new validity period that *excludes* revocation. This procedure is performed by the Root CA of a PKI, yielding a *cleaned PKI*, which has neither revocation nor intermediary CAs. The Root CA also *authenticates* all the timestamps issued by the subordinate Timestamp Authorities. A digital signature in the cleaned PKI is *trustworthy* as long as the used cryptographic algorithms remain secure. The proposal benefits archivers (e.g. digital libraries), whose computational effort to preserve trustworthy signatures is reduced. Yet, the users of signed documents are benefited too, because the processing time to evaluate signatures is reduced.

Organization In Section 2, we show that preserving and verifying digital signatures in the long term are hard tasks. In this context, we state the problems that our proposal addresses. We provide the related work in Section 3. We present a top-level description of our proposal, formal definitions and an illustrated example in Section 4. Section 5 describes timestamps, detailing the problems stated in Section 2 and also providing a solution. Section 6 presents the procedures and players in our scheme. Section 7 presents the benefits of the cleaned PKI. Finally, we present our considerations and future work (Section 8).

2. Problems of Signatures in the Long Term

Public Key Infrastructure (PKI) [Kohnfelder 1978] is a well-known means to ensure authenticity, integrity and proof-of-existence for digital documents. Authenticity and integrity can be guaranteed by digital signatures. The verification of digital signature requires trustworthy public keys, which are provided by a PKI. Proof-of-existence is guaranteed by timestamps, whose verification may require trustworthy public keys too.

The X.509 [ITU-T 2005] is the most prominent model of PKI, in which entities' public keys are organized in a hierarchy. On the highest level there is the Root CA, bellow which there are other Certification Authorities (CA), Timestamp Authorities (TSA) and final users. The X.509 technically describes public key certificates, and revocation status (e.g. Certificate Revocation List). From here on, we refer to X.509 as the *conventional PKI*.

A trustworthy signature requires mathematical and semantic correctness. The mathematical correctness is guaranteed if the signature is valid under the signer's public key. The semantic correctness is ensured if the binding between the signer's public key and his credentials, i.e. his public key certificate, *was* valid *when* the signature was created. Since the time when a signature was created is hard to securely guarantee by digital means, the time reference of the binding validity is shifted to the present. Thus, the semantic correctness is guaranteed if: a) the signer's certificate and corresponding superordinate CAs' certificates have not expired and have not been revoked *so far*; and b) the used cryptographic algorithms are *currently* secure. The first condition is verified with the help of the so-called *certification path validation* algorithm [Cooper et al. 2008]. The verification of the second condition requires the knowledge of the current status of cryptographic algorithms. Such information has been provided by governmental agencies, e.g. [National Institute of Standards and Technology 2007]

The semantic correctness is easily verified for short-term signatures, when the signing and verification times are close. This is because the signer's and superordinate CAs' certificates have probably not expired yet, revocation status is up to date and available, and the cryptographic algorithms are still secure. Nevertheless, this situation changes in the long term. Therefore, proof-of-existence is necessary to ensure that all those requirements were fulfilled at a time in the past (preferably as close as possible to the signing time). Signed timestamps [Adams et al. 2001a] are the most used means for proof-of-existence in the conventional PKI. Thus, certificates, their revocation status and timestamps are necessary to evaluate the semantic correctness of a signature in the long term. From here on, we refer to these objects as the *protection data* of a signed document.

Since timestamps are signed, the semantic correctness of their signatures should

be also verified. Again, proof-of-existence is necessary to allow such verification in the long term. As consequence, new timestamps are required, being the preservation of trustworthiness signatures an end-less process. In this context, we point out the following problems.

Problem 1. *The effort to preserve signed documents is not optimal*. Given a set of documents that were signed by different signers, preserving this set usually requires the archiver a non-optimal effort. The reason is that the archiver has to monitor the trustworthiness of each document's signature, applying new timestamps on it *before* the signer's and timestamper's certificates expire (for a detailed explanation, see Section 5). Depending on the number of different certificates and corresponding expiration dates, the archiver is likely to schedule timestamping procedures on several dates, instead of accessing the archive *as few as possible*. Therefore, this approach is neither optimal nor scalable [Huhnlein et al. 2009].

Problem 2. *The amount of protection data to store and verify is not optimal in the long term.* The regular use of timestamps on a signed document increases the protection data. The reason is that a timestamp is a signed document too, therefore protection data for it is necessary. Thus, the older the signed document is, the more timestamps are needed, the more protection data exists, the more storage space is used. Moreover, the more protection data exists, the more signatures a verifier has to evaluate to infer the trustworthiness of a signed document.

3. Related Work

In literature there are many proposals to improve the conventional PKI. We present them in regard to the problems 1 and 2.

Evidence Record Syntax (ERS) [Gondrom et al. 2007] is a solution that addresses Problem 1. Given a set of signed documents, it is timestamped by a single timestamp as follows. Each signed document is hashed. The digests are leaves of a Merkle Tree, whose root is timestamped by a Timestamp Authority (TSA). A copy of the timestamp is stored with each signed document. The archiver has to monitor only the lifetime of the TSA's certificate and the security of the used cryptographic algorithms. Although ERS addresses Problem 1, Problem 2 remains because the demand of timestamps is not reduced.

The amount of protection data is proportional to the length of a certification path [Martinez-Pel et al. 2008]. In that sense, *Nested Certificates* [Levi et al. 2004] address Problem 2 by reducing certification paths. Given a certification path C of length n, a *shortcut* is created between two certificates $c_i, c_j \in C$, such that $0 \leq i < j + 1 < n$. The shortcut is a new certificate issued by the subject of c_j and it asserts that the certificates $c_i, c_{i+1}...c_{j-1}$ are trustworthy. Although attractive, this approach creates additional trusted entity. That is, using a nested certificate requires the trust in the shortcut issuer.

Considering that Certificate Revocation List (CRL) is the usual data structure to implement revocation in the conventional PKI, protection data can be reduced by using schemes that are more efficient than CRL in regard to Problem 2. A straightforward approach is to eliminate the need for revocation by using *Short-Term Certificates* [Rivest 1998]. That is, certificates that are trustworthy for short periods (e.g. a day),

so that their probability of being revoked is almost zero. Nevertheless, those certificates are not practical for a CA because once the CA's certificate expires, all subordinate certificates have to be reissued. Moreover, *Online Certificate Status Protocol* (OCSP) [Myers et al. 1999] and *Novomodo* [Micali 2002] provide revocation data that is smaller and easier to evaluate than CRL. These solutions just mitigate Problem 2, because there is still the accumulation of timestamps and certificates. Yet, Problem 1 remains.

Solutions as [Moecke et al. 2010, Vigil et al. 2009, Adams et al. 2001b] use the help of an additional trusted entity. It verifies the semantic and mathematical correctness of a document's signature. If the verification succeeds, the trusted entity asserts that the signature is trustworthy. The assertion is signed by the trusted entity. The original protection data of the signed document is replaced by only the signed assertion and the trusted party's certificate. This replacement allows to address Problem 2. Problem 1 remains. Yet, the addition of a trusted entity creates a further point of failure in the PKI.

4. Cleaning the PKI up

In a conventional PKI, a Certification Authority (CA) asserts that the binding between a public key and an entity is trustworthy. This assertion is conveyed by a certificate. The CA defines a validity period for the certificate, after which the certificate expires and is no longer trustworthy. However, the certificate can be revoked at *any time* during the validity period. This possibility leads to the problems 1 and 2. To address them, we propose to *clean up the PKI*, as describe hereinafter. For each description, we provide a formal definition in Section 4.1. Moreover, we give an illustrated example of our proposal in Section 4.2

Given a conventional PKI (cf. Definition 1), which provides trustworthy certificates for the use of signed documents (cf. Definition 4), we clean the PKI up, yielding a *cleaned PKI*. To do so, the Root CA of the conventional PKI reissues each final certificate, but defining a *new* validity period during which the certificate *was indeed trustworthy*. The outcome of this process (cf. Definition 3) is a *cleaned PKI* (cf. Definition 2).

Our proposal addresses Problem 2 in the following way. Since the Root CA reissues final certificates of the conventional PKI, the intermediary CAs' certificates are not present in the cleaned PKI. Yet, the Root CA redefines the validity period of each final certificate in the cleaned PKI, excluding the period *after* the revocation of the certificate in the conventional PKI. For example, if a certificate was valid for 365 days and was revoked on the 50^{th} day, then the Root CA reissues the certificate using a validity period of 49 days. Since intermediary CAs' certificates and revocation data are no longer necessary after cleaning the PKI up, the size of *protection data* in a signed document is reduced, thereby addressing Problem 2.

We address Problem 1 by reissuing the certificate of a Timestamps Authority (TSA) as a final certificate, but applying the idea of *Off-line Timestamps* [Ansper et al. 2001]. That is, given a TSA and the timestamps that it has ever issued, the Root CA binds the timestamps with the reissued TSA's certificate. This binding *authenticates* the trustworthy timestamps, while preventing forged timestamps, which were created *after* the TSA expired, from being accepted as trustworthy. As a consequence, an archiver only retimestamps a signed document *before* the used cryptographic algorithm become insecure. This reduces the effort of the archiver, addressing Problem 1. For a

detailed explanation, we refer the reader to Section 5.

4.1. Definitions

In the following, we formally define the concepts involved in the cleaned PKI. Note that we use *tuples* to define certificates, signed documents and protection data. Moreover, given a tuple Z of n elements, we denote the k^{th} element as $\pi_n(Z)$, such that $1 \leq k \leq n$.

Definition 1. A *conventional PKI* is a directed graph $G = (V, A)$. The set V comprises the public keys p_i, being $i \geq 0$ and p_0 assigned to the Root CA. The set A comprises certificates $c_{j,k} = (p_j, p_k)$, such that $p_j, p_k \in V$ and the subject of p_j creates (i.e. signs) $c_{j,k}$ for the subject of p_k. The graph G is almost a tree, since there is a single cycle $c_{0,0}$ (a self-signed certificate for the Root CA). Each certificate $c_{j,k} \in A$ is qualified by a time interval $v_{j,k} = [i_{j,k}, e_{j,k}]$, being $i_{j,k}$ and $e_{j,k}$ the issuance and expiration dates for $c_{j,k}$ respectively, such that $i_{j,k} < e_{j,k}$. Moreover, given a certificate and a corresponding sibling, namely $c_{j,k}, c_{k,l} \in A$, then $v_{j,k} \cap v_{k,l} = v_{k,l}$. Furthermore, $c_{j,k} \in A$ *can* be revoked on a date $r_{j,k}$. If $c_{j,k}$ is revoked, then $r_{j,k} \in v_{j,k}$ and the corresponding descendants certificates $c_{t,u} \in A$ become *invalid* from $r_{j,k}$ on. Otherwise, $r_{j,k} > e_{j,k}$.

Definition 2. A *cleaned PKI* is a *conventional PKI* $G' = (V', A')$ (cf. Definition 1), such that $\forall c_{j,k} \in A'$, $j = 0, k \geq 0$. That is, all certificates are issued by the Root CA. Moreover, a certificate $c_{0,k} \in A'$ *cannot* be revoked anytime, i.e. it is always valid in $v_{0,k}$. Therefore, we eliminate the revocation dates $r_{j,k}$ from G'.

Definition 3. A *clean-up function* is a function $f_c : G \mapsto G'$, such that $G = (V, A)$ is a conventional PKI (cf. Definition 1) and $G' = (V', A')$ is a cleaned PKI (cf. Definition 2). The function f_c receives as input G and a date u, such that $u \in v_{0,0}$. The function f_c works as presented in Algorithm 1. Line 11 prevents a forged certificate, i.e. a certificate created *after* the revocation of any of its ancestors. In line 12, a signature scheme is implicit in the creation of $c_{0,k}$, being more details given in Section 6.2. Note that $A' \cap A = c_{0,0}$ and $V' = V - \{p_i\}$, such that $p_i \in V$ and corresponds to an intermediary CA.

Definition 4. A *signed document* is a tuple $S_{d,i} = (d, \sigma_{d,i}, P_i)$, being d a data object whose signature $\sigma_{d,i}$ is verifiable under public key $p_i \in V$, such that $G = (V, A)$ is a graph that represents a PKI (cf. definitions 1 and 2). The protection data is a tuple $P_i = (c_{k,i}, I_i, R_i, T_i)$ that comprises the certificate $c_{k,i} \in A$ for p_i, the set $I_i \subset A$ of certificates that are ancestors of $c_{k,i}$, the set R_i of revocation status for $c_{k,i}$ and each certificate in I_i, a set T_i of signed timestamps $S_{n,p}$, such that $T_p = \emptyset$. That is, a timestamp is a signed document that does not encapsulate further timestamps.

4.2. Example

Figure 1 depicts an example in which a conventional PKI G (Figure 1(a)) is cleaned up, yielding the cleaned PKI G' (Figure 1(b)). In G there are a Root CA's certificate ($c_{0,0}$), intermediaries CAs' certificates ($c_{0,1}$ and $c_{0,2}$) and the final certificates ($c_{1,3}$,$c_{1,4}$,$c_{1,5}$,$c_{2,6}$,$c_{2,7}$ and $c_{2,8}$). The certificates $c_{0,1}$ and $c_{2,7}$ are revoked (see $\square$symbol) on dates $r_{0,1}$ and $r_{2,7}$ respectively.

On a date u, G is cleaned up, yielding G'. Note that in G' there is *neither* intermediary CA *nor* revocation. Also, see that a final certificate in G' may have a shorter validity

Input: A conventional PKI $G = (V, A)$, a set T of digests, and a date u
Output: A cleaned PKI $G' = (V', A')$

$V' \leftarrow \{p_0\}$, such that $p_0 \in V$
$A' \leftarrow \{c_{0,0}\}$, such that $c_{0,0} \in A$
$G' \leftarrow (V', A')$
foreach *leaf* $c_{j,k} \in A$, *such that* $i_{j,k} < u$ **do**
    $R \leftarrow \{c\}$
    $R \leftarrow R \cup \{min(e_{j,k}, r_{j,k})\}$
    foreach *ancestor* $c_{l,m}$ *of* $c_{j,k}$ **do**
        $R \leftarrow R \cup \{min(e_{l,m}, r_{l,m})\}$
    end
    $e_{0,k} \leftarrow min(R)$
    if $i_{j,k} < e_{0,k}$ **then**
        $c_{0,k} \leftarrow (p_0, p_k)$, such that $p_0, p_k \in V$
        $v_{0,k} \leftarrow [i_{j,k}, e_{0,k})$
        if p_k *belongs to a TSA* **then**
            associate $c_{0,k}$ to $\lambda_k \in T$
        end
        $A' \leftarrow A' \cup \{c_{0,k}\}$
    end
end
return G'

Algorithm 1: The clean-up function f_c.

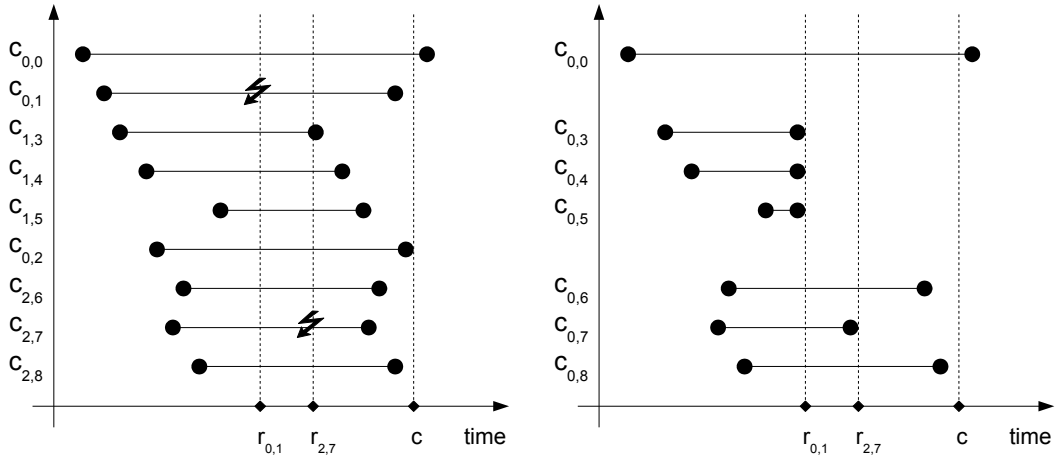
period than the corresponding final certificate in G, depending on a revocation in G. That is, if a certificate in G is revoked, it or its descendants is/are reissued in G', but using a shorter validity period that excludes the period after the revocation. For instance, $c_{0,1}$ is revoked on date $r_{0,1}$ (Figure 1(a)), therefore $c_{0,3}, c_{0,4}, c_{0,5}$ (Figure 1(b)) have shorter validity periods than $c_{1,3}, c_{1,4}, c_{1,5}$ (Figure 1(a)). Yet, note that the certificates of intermediary CAs (namely $c_{0,1}$ and $c_{0,2}$) are not reissued, therefore they are not present in Figure 1(b).

5. Timestamps

Timestamps are the most prominent means to guarantee proof-of-existence for data objects in a PKI. That is, timestamps allow us to infer a date on which a data object existed. This facility is necessary to sort data objects in chronological order. For instance, given the compromise of a private key k_p on date r_p and a signature σ_p done by k_p, a timestamp on σ_p on a date t_{σ_p} allows us to check if σ_p is trustworthy, i.e. if σ_p was created *before* the compromise of k_p. In other words, if $t_{\sigma_p} < r_p$.

The use of timestamps requires the so-called Timestamp Authority (TSA), which is a trusted entity that provides *signed* timestamps with a *trustworthy* date value. That is, given a data object, the TSA provides a timestamp that binds the data object to the *current* value of the TSA's clock, which is protected and synchronized with a legal source of time.

The compromise of the private key of a TSA invalidates all its issued timestamps.



(a) A conventional PKI G in which certificates c_1 and c_7 are revoked at $r_{0,1}$ and $r_{2,7}$ respectively.

(b) A cleaned PKI G', yielded by cleaning G up.

Figure 1. Comparison between conventional and cleaned PKIs.

The reason is that an adversary is able to forge timestamps on behalf of the compromised TSA using *any* date value. As a countermeasure, the CA that issued the certificate of the compromised TSA notices all verifiers that the TSA is no longer trustworthy. That is, the TSA is revoked.

A CA is liable for the revocation of the certificates that the CA issues, but *until* these certificates expire. From this fact Problem 1 raises. For instance, an adversary may compromise the private key of a TSA *after* it goes out of business. In this case, the adversary is able to forge timestamps and verifiers cannot notice the forgery because the TSA's certificate is expired and no revocation status is available. To prevent such a forgery, we *timestamp* timestamps, a process that is end-less for the archiver of signed documents.

A solution to avoid Problem 1 is the destruction of the TSA's private key after the TSA finishes its operations. Nevertheless, proving that a data object was *properly* erased is still an open problem. Therefore, we *assume* that the key is properly erased. Given the number of possible TSAs in a PKI and respective security policies, this trust assumption *may* be prohibitive. Thus, we propose to reduce our assumption to the private key of the Root CA, which authenticates all timestamps issued *before* the PKI is cleaned up. Our proposal uses the idea of *Off-line Timestamps* [Ansper et al. 2001] and is detailed in the following.

Let $S_{n,p}$ be timestamps (cf. Definition 4) issued by a TSA whose certificate $c_{k,p} \in A$, such that $G = (V, A)$ is a conventional PKI (cf. Definition 1) and $n \geq 0$. Before a date u, when G is cleaned up, the TSA builds a Merkle Tree [Merkle 1989] whose leaves are the digest of each $S_{n,p}$ and λ_p is the root. Afterwards, the Root CA executes the clean up function f_c (cf. Definition 3) in order to create the cleaned PKI $G' = (V', A')$ (cf. Definition 2). While issuing the certificate $c_{0,p} \in A'$ for the TSA (cf. line 15, Algorithm 1), the Root CA binds λ_p to $c_{0,p}$. Since $c_{0,p}$ is dated by a validity period $v_{0,p} = [i_p, e_p)$, λ_p and consequently all $S_{n,p}$ are timestamped. That is, we infer that

$S_{n,p}$ existed *before* the date i_p if we can reconstruct λ_p using the digest of $S_{n,p}$ and the corresponding *authentication path* to λ_p.

Given a forged timestamp $S_{n+1,p}$ created on a date $t_f > i_p$, an adversary is not able to prove that $S_{n+1,p}$ is trustworthy. That is, the adversary is not able to find out a digest for $S_{n+1,p}$ and authentication path that allow him to reconstruct λ_p as long as the cryptographic algorithms used to create λ_p and $c_{0,p}$ are secure.

Note that we extended the trustworthiness of a timestamp beyond the expiration of the corresponding TSA's certificate, being now the lifetime of the used cryptographic algorithms the constraint. Thus, the archiver needs to monitor only the security of the cryptographic algorithms instead of the expiration of certificates. Since the number of the used cryptographic algorithms is *likely* to be smaller than the number of certificates in an archive, the effort of the archiver is reduced, addressing Problem 1. Furthermore, the lifetime of cryptographic algorithms *usually* lasts longer than the lifetime of final certificates (e.g. in ICP-Brasil). Thus, Problem 2 is addressed, since less timestamps are necessary, requiring then the storage and verification of less protection data.

6. Procedures

The procedures described in this section show how the definitions given in sections 4 and 5 are applied in a practical sense, involving the following players. The Root CA gathers certificates, revocation and timestamps (cf. Section 6.1). Next, the Root CA reissues final certificates and authenticates timestamps (Section 6.2), yielding a cleaned PKI. Given the certificates in a cleaned PKI, archivers replace certificates and revocation data within signed documents (cf. Section 6.3). A final user that uses archived and signed documents evaluates their trustworthiness by verifying their signatures (cf. Section 6.4).

6.1. Preparing to Clean up

The preparation for cleaning a conventional PKI up is performed by the Root CA and consists of two steps: a) gather all issued certificates in the PKI and the current revocation for each certificate; b) collect a representation of the issued timestamps in the conventional PKI; and c) verify the mathematical correctness of the signatures on the gathered certificates and revocation status. The outcome of executing this protocol is a graph $G = (V, A)$ representing all public keys and certificate in the conventional PKI (cf. Definition 1). In the following, we detail each step.

Gathering the issued certificates and revocation status requires visiting the repository of each Certification Authority in the PKI, starting from the Root CA. This is a well-known task that can be automated, therefore we just assume it is already implemented in a PKI and it returns a graph $G = (V, A)$ and the corresponding revocation status for each gathered certificate.

Collecting a representation of the issued timestamps in the conventional PKI represented by $G = (V, A)$ consists of the following. From each TSA whose certificate $c_{k,i} \in A$, the Merkle Tree's root λ_i that represents the timestamps the TSA has ever issued (cf. Section 5) is queried. We assume this task is also implemented in the PKI and returns a set T of Merkle Trees' roots, such that $\lambda_i \in T$

Since a certificate is a signed object, the mathematical correctness of the certificate's signature is necessary, but not sufficient, to ensure the certificate authenticity.

Therefore, for each certificate $c_{j,k} \in A$, the signature on $c_{j,k}$ should be valid under the public key $p_j \in V$. A revocation status is usually signed, depending on how its authenticity is ensured. Considering that the revocation status $r_{j,k}$ for certificate $c_{j,k}$ is signed, then the signature on $r_{j,k}$ should be valid under the public key p_j.

Note that the semantic correctness for each $c_{j,k} \in A$ is not evaluated here, since the deployment of the clean-up function f_c (cf. Definition 3) includes this task. That is, f_c verifies validity periods and revocation status.

By finishing the above described steps, this procedure outcomes a graph $G = (V, A)$ whose certificates' mathematically correctness is ensured. This result is used in next procedure, which cleans G up (cf. Section 6.2).

6.2. Reissuing Final Certificates

Given a graph $G = (V, A)$ representing the certificates in a conventional PKI gathered and partially verified in the preparation procedure and the set T of Merkle Trees' roots that resumes the timestamps issued in the conventional PKI (cf. Section 6.1), the Root CA executes the reissuing procedure. This procedure outcomes a cleaned PKI $G' = (V', A')$ (cf. Definition 2) and consists of executing the clean-up function f_c (cf. Definition 3), using G and an arbitrary date u as inputs. In the following, we detail u and f_c.

Recall that u defines the time reference for cleaning a PKI up, being relevant for revocation status and limiting the validity period of a reissued certificate. The choice of different values for u and the corresponding implication are discussed in Section 8.

After u is chosen, f_c is executed by the Root CA. Note that the Root CA can be overloaded, depending on the number of final certificates in G that will be reissued (cf. Algorithm 1). Therefore, we propose the use of Merkle Trees to optimize the signature of final certificates.

Instead of signing each certificate $c_{0,k} \in A'$, such that $G' = (V', A')$ is a cleaned PKI and $c_{j,k} \in A$ is a leaf in a conventional PKI $G = (V, A)$, the Root CA creates $c_{0,k}$ without a signature. If $p_k \in V$ is the public key of a TSA, $\lambda_k \in T$ is associated to $c_{0,k}$. After f_c returns G', a Merkle Tree is built, being $\lambda_{A'}$ the root and the leaves the digest of each certificate $c_{0,k} \in A'$. Then, a single signature $\sigma_{0,\lambda_{A'}}$ on $\lambda_{A'}$ is done by the Root CA, using its private key. Next, for each $c_{0,k}$ an authentication path to $\lambda_{A'}$ is calculated and attached to $c_{0,k}$ along with $\lambda_{A'}$.

Being all certificates in G' signed, archivers can use them to replace the certificates of G in a signed document. This procedure is described in Section 6.3.

6.3. Replacing Protection Data

The replacement of protection data is performed by an archiver for each signed document. This procedure consists of generating new and reduced protection data, which replaces the existing protection data in a signed document. In the following, we detail this procedure.

Given a conventional PKI $G = (V, A)$ (cf. Definition 1), consider that an archive comprises signed documents $S_{d,i} = (d, \sigma_{d,i}, P_i)$ (cf. Definition 4), being $P_i = (c_{k,i}, I_i, R_i, T_i)$ the protection data and $c_{k,i} \in A$ a signer's certificate. Moreover, there may exist signed timestamps $S_{S_{d,i},p} = (S_{d,i}, \sigma_{S_{d,i},p}, P_p)$ (cf. Definition 4) on $S_{d,i}$, being

Input: A protection data $P_i = (c_{k,i}, I_i, R_i, T_i)$, a conventional $G = (V, A)$ and cleaned $G' = (V', A')$ PKIs
Output: A cleaned protection data $P'_i = (c_{0,i}, I'_i, R'_i, T'_i)$

1 $T'_i \leftarrow \emptyset$
2 **if** $T_i \neq \emptyset$ **then**
3 $\quad S_{S_{d,i,p}} \leftarrow \pi_1(T_i)$
4 $\quad H_{\lambda_p, S_{S_{d,i,p}}} \leftarrow$ get auth. path from the digest of $S_{S_{d,i,p}}$ to λ_p
5 $\quad S_{S_{d,i,p}} \leftarrow S_{S_{d,i,p}} || H_{\lambda_p, S_{S_{d,i,p}}}$
6 $\quad P_p \leftarrow \pi_4(S_{S_{d,i,p}})$
7 $\quad P'_p \leftarrow$ execute Algorithm 2 using inputs P_p, G, and G'
8 $\quad \pi_4(S_{S_{d,i,p}}) \leftarrow P'_p$
9 $\quad T'_i \leftarrow T'_i \cup \{S_{S_{d,i,p}}\}$
10 **end**
11 $P'_i \leftarrow (c_{0,i}, \emptyset, \emptyset, T'_i)$, such that $c_{0,i} \in A'$
12 **return** P'_i

Algorithm 2: Creating a cleaned protection data.

$S_{S_{d,i,p}} \in T_i$, $P_p = (c_{j,p}, I_p, R_p, T_p)$ the protection data and $c_{j,p} \in A$ a TSA's certificate. Also, $c_{k,i}$ and $c_{j,p}$ are final certificates, i.e. $k \neq i$, $j \neq p$ and $k, i, j, p > 0$.

After G is cleaned up, a cleaned PKI $G' = (V', A')$ (cf. Definition 2) is obtained. Then, the archiver executes Algorithm 2, using the protection data P_i, G, and G'. The algorithm returns new protection data P'_i. Then, the archiver replaces P_i by P'_i in the signed document $S_{d,i}$.

Note that Algorithm 2 removes timestamps in T_i, but keeps the first timestamp. The reason is that multiple timestamps address certificate expiration, which is not an issue in the cleaned PKI (cf. Section 5). Therefore, the first timestamp is able to provide the signed document $S_{d,i}$ with proof-of-existence until the used cryptographic algorithms remain secure. We pointed out that Algorithm 2 does not consider a timestamp issued by a TSA in another conventional PKI $G'' \neq G$, however the algorithm can be easily adapted to do so. This is necessary in the long term, after the cryptographic algorithms are no longer secure in G' and a further timestamp under a new PKI, which uses up to date cryptographic algorithms, is required.

The Root CA signs the Merkle Tree's root λ_p, which represents the authenticated timestamps $S_{S_{d,i,p}}$ (cf. Section 6.2). To evaluate the trustworthiness of $S_{S_{d,i,p}}$, verifiers need the authentication path $H_{\lambda_p, S_{S_{d,i,p}}}$ from the digest of $S_{S_{d,i,p}}$ to λ_p. To do so, there are lines 4 and 5 in Algorithm 2.

Protection data P'_i requires less storage space than P_i, because P'_i comprises neither intermediary CAs' certificates nor revocation information. That is, $I_i, R_i = \emptyset$. Moreover, we restrict the set T'_i to a single timestamp, which also has a reduced protection data P'_p. By these reductions, we address problems 1 and 2.

The replacement of P_i by P'_i preserves the signer's and TSA's public keys. Also, the Root CA is the same in both PKIs. As a consequence, this procedure does not prevent

the verification of signatures, as described in Section 6.4.

6.4. Verifying a Signed Document

The verification of a signed document is performed by a final user who wants to evaluate whether the content in the signed document is trustworthy. This procedure consists of evaluating the signatures in the signed document and corresponding timestamps, by checking the mathematical and semantic correctness of each signature (cf. Section 2). In the following, we detail this procedure.

Consider a signed document $S_{d,i} = (d, \sigma_{d,i}, P'_i)$ (cf. Definition 4), being d the content whose trustworthiness is evaluated, $\sigma_{d,i}$ the signature on d, $P'_i = (c_{0,i}, \emptyset, \emptyset, T'_i)$ the *reduced* protection data, $c_{0,i} \in A'$ the signer's certificate, such that $G' = (V', A')$ is a cleaned PKI (cf. Definition 2), $T'_i = \{S_{S_{d,i},p}\}$ a set, $S_{S_{d,i},p} = (S_{d,i}, \sigma_{S_{d,i},p}, P'_p)$ (cf. Definition 4) a timestamp on $S_{d,i}$, $P'_p = (c_{0,p}, \emptyset, \emptyset, \emptyset)$ the *reduced* protection data and $c_{0,p} \in A'$ a TSA's certificate. Also, $c_{0,i}$ and $c_{0,p}$ are final certificates, i.e. $i \neq p$ and $i, p > 0$. There are the Merkle Tree's root $\lambda_{A'}$ associated to $c_{0,i}$ and $c_{0,p}$, and the authentication paths $H_{\lambda_{A'},i}$ and $H_{\lambda_{A'},p}$ associated to $c_{0,i}$ and $c_{0,p}$ respectively (cf. Section 6.2). Also, there are the Merkle Tree's root λ_p associated to $c_{0,p}$ and an authentication path $H_{\lambda_p,S_{S_{d,i},p}}$ from the digest of $S_{S_{d,i},p}$ to λ_p attached to $S_{S_{d,i},p}$ (cf. Section 6.3).

To verify d, the signature $\sigma_{d,i}$ is evaluated in terms of mathematical and semantic correctness. The mathematical verification consists of checking whether $\sigma_{d,i}$ is valid under p_i, being p_i available in $c_{0,i}$. The semantic verification checks whether $c_{0,i}$ was valid on the date t_p stated by the timestamp $S_{S_{d,i},p}$. This is ensured if the following conditions hold:

a) the signature on $c_{0,i}$ is valid under p_0;
b) the signature on $c_{0,p}$ is valid under p_0;
c) $\sigma_{S_{d,i},p}$ is valid under $p_{0,p}$;
d) $v_{0,i} \cap v_{0,0} = v_{0,i}$;
e) $v_{0,p} \cap v_{0,0} = v_{0,p}$;
f) $S_{S_{d,i},p}$ is *authenticated* by the Root CA in $c_{0,p}$; and
g) the timestamp date $t_p \in v_{0,i}$.

Note that in conditions a) to c) we verify the mathematical correctness of the signature on the certificates $c_{0,i}$ and $c_{0,p}$ and the timestamp $S_{S_{d,i},p}$, ensuring that these objects are not corrupted. Recall that a) and b) use the Merkle Tree $\lambda_{A'}$ and the authentication paths $H_{\lambda_{A'},c_{0,i}}$ and $H_{\lambda_{A'},c_{0,p}}$. Conditions d) and e) guarantee that the validity periods of $c_{0,i}$ and $c_{0,p}$ do not go beyond the lifetime of Root CA's certificate $c_{0,0}$. Conditions f) and g) prevent forgeries if the signer's or TSA's private key is compromised after the corresponding certificate is expired. Recall, that f) uses the Merkle Tree's root λ_p the authentication path $H_{\lambda_p,S_{S_{d,i},p}}$.

7. Benefits

By switching from the conventional PKI to the cleaned PKI, we address problems 1 and 2. As a consequence, we simplify the preservation and use of signed documents, thereby benefiting the archivers and final users respectively.

While in the conventional PKI the semantic correctness of a signature depends on certificates' validity periods, in the cleaned PKI only the security of cryptographic algorithms is important. The lifetime of a final certificate is *usually* shorter than cryptographic algorithms' lifetimes. Taking the Brazilian National PKI (a.k.a ICP-Brasil) as an example, a final certificate expires in 3 years [ITI 2008b] and the recommended signature suite SHA-256 with RSA-2048 is expected to be secure for 16 years [ITI 2008a] from now on. Since semantic correctness lasts longer in the cleaned PKI, less timestamps are used than in the conventional PKI in the long term. This benefits the archiver because he needs to request fewer timestamps. Yet, it is easier for the archiver to monitor some cryptographic algorithms' lifetimes than several certificates' lifetimes.

The fewer timestamps, the smaller protection data. This benefits the archiver by reducing the costs with storage space. The smaller data protection, the fewer signatures have to be evaluated (e.g. on timestamps, certificates and revocation data). Therefore, final users are benefited because less processing time is necessary to verify the trustworthiness of a signed document. Yet, the absence of revocation eliminates the need to download fresh revocation data each time a signed document is verified. As a consequence, final users can perform *off-line* verifications, i.e. without network access.

Unlike some related works (e.g. [Levi et al. 2004, Vigil et al. 2009, Adams et al. 2001b]), our proposal do not add an extra *trusted* entity, which would strengthen the trust assumption of a conventional PKI. The cleaned PKI has only two failure points, namely the Root CA and the cryptographic algorithms. That is, unless both remain secure (e.g. the Root CA is not compromised and the cryptographic algorithms are not suddenly broken), the cleaned PKI is secure.

8. Considerations & Future Work

All in all, we propose a new approach that *cleans up* the conventional PKI to facilitate the use of long-term signatures. We suggest removing intermediary CAs and revocation. To do so, the Root CA reissues final certificates defining validity periods that exclude revocation. The Root CA authenticates all trustworthy timestamps ever issued by any subordinate TSA. Reissued certificates remain trustworthy as long as the used cryptographic algorithms are secure. In the cleaned PKI, archivers use fewer timestamps than in the conventional PKI. Timestamps are only necessary to renew cryptographic algorithms. As a benefit, we reduce: a) the effort to preserve signed documents (Problem 1); and b) the storage space and processing time to handle signed documents (Problem 2). Yet, the cleaned PKI adds no further *trusted entity*.

The overall procedure of cleaning up a PKI is expensive in terms of processing time. The reasons are: a) for each final certificate, the mathematical and semantic correctness are verified; and b) a Merkle Tree, whose leaves are the digest of each final certificate, is created. The execution of both tasks can be delegated to multiple computers while the Root CA is off-line. We point out that the Root CA's private key is used once to sign the Merkle Tree's root. The Root CA does not sign each final certificate.

The replacement of protection data is prevented in the current signature schemes, e.g. XAdES [ESI 2008]. The reason is that these schemes force the signer the sign the purposed data object along with his certificate. Although the certificate can be replaced later, the mathematical correctness of the signature will be compromised.

As future work, we suggest to work out the issue of replacing validation data. For instance, signature schemes could bind the signer's meta-data instead of the certificate to the signature, thereby allowing replacements. Another possibility is to explore the use of chameleon hash functions to replace certificates based on collisions [Le et al. 2008].

Acknowledgements

The authors thank Daniel Cabarcas and the anonymous reviewers for the comments that helped to improve this manuscript. Also, the authors thank Coordenação de Aperfeiçoamento de Pessoal de Nível Superior for supporting this work.

References

[Adams et al. 2001a] Adams, C., Cain, P., Pinkas, D., and Zuccherato, R. (2001a). Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161 (Proposed Standard).

[Adams et al. 2001b] Adams, C., Sylvester, P., Zolotarev, M., and Zuccherato, R. (2001b). Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols. RFC 3029 (Experimental).

[Ansper et al. 2001] Ansper, A., Buldas, A., Roos, M., and Willemson, J. (2001). Efficient long-term validation of digital signatures. In *Public Key Cryptography*, pages 402–415. Springer.

[Cooper et al. 2008] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and Polk, W. (2008). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard).

[ESI 2008] ESI (2008). Electronic Signatures and Infrastructures (ESI); Profiles of XML Advanced Electronic Signatures based on TS 101 903 (XAdES). Technical report, European Telecommunications Standards Institute.

[Gondrom et al. 2007] Gondrom, T., Brandner, R., and Pordesch, U. (2007). Evidence Record Syntax (ERS).

[Gutmann 2002] Gutmann, P. (2002). Pki: it's not dead, just resting. *Computer*, 35(8):41–49.

[Huhnlein et al. 2009] Huhnlein, D., Korte, U., Langer, L., and Wiesmaier, A. (2009). A comprehensive reference architecture for trustworthy long-term archiving of sensitive data. In *New Technologies, Mobility and Security (NTMS), 2009 3rd International Conference on*, pages 1–5. IEEE.

[ITI 2008a] ITI (2008a). *Declaração de Práticas de Certificação da Autoridade Certificadora Raiz da ICP-Brasil*. Instituto Nacional de Tecnologia da Informação, Brasília, v.4.0 edition. DOC-ICP-01.

[ITI 2008b] ITI (2008b). *Requisitos Mínimos para as Políticas de Certificado na ICP-Brasil*. Instituto Nacional de Tecnologia da Informação, Brasília, v.3.0 edition. DOC-ICP-04.

[ITU-T 2005] ITU-T (2005). Recommendation X.509 (08/2005) - Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks. Technical report, ITU-T.

[Kohnfelder 1978] Kohnfelder, L. M. (1978). Towards a practical public-key cryptosystem. Technical report, Massachusetts Institute of Technology.

[Le et al. 2008] Le, D., Bonnecaze, A., and Gabillon, A. (2008). A secure round-based timestamping scheme with absolute timestamps (short paper). *Information Systems Security*, pages 116–123.

[Levi et al. 2004] Levi, A., Caglayan, M., and Koc, C. (2004). Use of nested certificates for efficient, dynamic, and trust preserving public key infrastructure. *ACM Transactions on Information and System Security (TISSEC)*, 7(1):21–59.

[Martinez-Pel et al. 2008] Martinez-Pel, R., Satiz, C., Rico-Novella, F., and Forn, J. (2008). Efficient Certificate Path Validation and Its Application in Mobile Payment Protocols. *2008 Third International Conference on Availability, Reliability and Security*, pages 701–708.

[Merkle 1989] Merkle, R. C. (1989). A Certified Digital Signature. In Brassard, G., editor, *Advances in Cryptology - CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer.

[Micali 2002] Micali, S. (2002). Scalable certificate validation and simplified pki management. In *1st Annual PKI Research Workshop*, page 15.

[Moecke et al. 2010] Moecke, C. T., Custódio, R. F., Kohler, J. G., and Carlos, M. C. (2010). Uma ICP baseada em certificados digitais autoassinados. In *Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 91–104, Fortaleza. SBSEG.

[Myers et al. 1999] Myers, M., Ankney, R., Malpani, A., Galperin, S., and Adams, C. (1999). X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 2560 (Proposed Standard).

[National Institute of Standards and Technology 2007] National Institute of Standards and Technology (2007). *Recommendation for Key Management – Part 1: General (Revised)*.

[Rivest 1998] Rivest, R. (1998). Can we eliminate certificate revocation lists? In *Financial Cryptography*, pages 178–183. Springer.

[Vigil et al. 2009] Vigil, M. A. G., da Silva, N., Moraes, R., and Custódio, R. F. (2009). Infra-estrutura de Chaves Públicas Otimizada: Uma ICP de Suporte a Assinaturas Eficientes para Documentos Eletrônicos. In *IX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg)*, Campinas.

