How to avoid the Breakdown of Public Key Infrastructures

Forward Secure Signatures for Certificate Authorities

Johannes Braun[1], Andreas Hülsing[1], Alex Wiesmaier[2], Martín A. G. Vigil[1], and Johannes Buchmann[1]

[1] Technische Universität Darmstadt
Hochschulstraße 10, 64283 Darmstadt, Germany
{jbraun,huelsing,vigil,buchmann}@cdc.informatik.tu-darmstadt.de
[2] AGT Group (R&D) GmbH
Hilpertstraße 20a, 64295 Darmstadt, Germany
awiesmaier@agtgermany.com

Abstract. Recent attacks and publications have shown the vulnerability of hierarchical Public Key Infrastructures (PKIs) and the fatal impact of revoked Certification Authority (CA) certificates in the PKIX validity model. Alternative validity models, such as the extended shell and the chain model, improve the situation but rely on independent proofs of existence, which are usually provided using time-stamps. As time-stamps are validated using certificates, they suffer from the same problems as the PKI they are supposed to protect. Our solution to this problem is abandoning time-stamps and providing proof of existence using Forward Secure Signatures (FSS). In particular, we present different possibilities to use the chain model together with FSS, resulting in schemes that include the necessary proofs of existence into the certificates themselves.

Keywords: PKI, CA, authentication, forward secure signature, revocation, certificate, time-stamping, validity model

1 Introduction

Public Key Infrastructures (PKIs) [22] are a well-known means of providing authenticity, non-repudiation, authentication, and datedness in electronic scenarios (e.g. e-commerce). The most prominent model of X.509 [19] PKIs consists of a hierarchy of Certification Authorities (CA) wherein X.509 certificates and time-stamps [1] are issued. Recent incidents involving Comodo [6], StartCom [17], and DigiNotar [18] showed that CAs are susceptible to attacks, requiring the revocation of CA certificates. But what happens upon revoking a CA's certificate $Cert_{CA}$ according to the shell model [7] used in X.509? First, as intended $Cert_{CA}$ can no longer be used for issuing certificates or revocations. Second, all formerly issued certificates and revocations relying on $Cert_{CA}$ become transitively invalid. Thus, leading to an immediate and complete breakdown of the spanned infrastructure as any authentication and signature verification relying on $Cert_{CA}$ will fail. This work proposes a solution to avoid this breakdown.

Contribution. We show how to provide durable protection against a PKI breakdown while abandoning time-stamps and the corresponding additional infrastructure. This is done by using Forward Secure Signatures (FSS) to include tamper proof time-tags into certificates to realize alternative validity models. Additionally, our solution can be used for implementing revocation on doubt policies, i.e. if there is the suspicion of a compromise, simplified revocation checking on client side, and fail-safe mechanisms.

Outline. The paper starts with an introduction to PKI and validity models. After that, we give a detailed problem statement followed by the solution: using the chain model with FSS. Then we describe implementation details and point out further advantages followed by related work and a comparison to our work. We end with a conclusion.

2 Background

In the following, we give a short introduction to PKIs and certificate validation.

2.1 Public Key Infrastructure in a Nutshell

A PKI supports the use of digital signatures [9] by handling keys and providing public key certificates. Certificates bind the signer's identity (e.g. a name) to his public key. The binding is certified by a Certification Authority (CA), which issues (i.e. signs) the certificate. CAs are often organized in a hierarchical structure. On the top exists a Root CA that issues certificates for Sub CAs. On the next level, Sub CAs issue certificates to lower Sub CAs or end-users.

The binding between a public key and an identity is ensured until the certificate either expires or is revoked. Expiration happens upon the end of the certificate's validity period, defined by the fields *NotBefore* and *NotAfter*. Revocation means the invalidation of a certificate during its validity period. For instance, when the identity changes or the private key is compromised. Revocation is done by the CA that certified the binding, often using Certificate Revocation Lists (CRL) [7] and / or the Online Certificate Status Protocol (OCSP) [26].

Datedness is another service supported by a PKI. Time-Stamp Authorities (TSA) provide this service by issuing time-stamps for given data digests (e.g. a document or a signature digest). A time-stamp is a signed object binding a given data digest to a trustworthy value of date and time.

2.2 Validity models

There are two aspects concerning the validity of electronic signatures. The first is the mathematical correctness of a signature, i.e. the signature is a valid signature under the given public key. The second aspect is the semantic correctness of a signature, i.e. the binding between the signer and the given public key is valid. In this work we are concerned with the semantic correctness and take the mathematical correctness for granted.

To verify the validity of a signature in a hierarchical PKI, the certification path from the Root Certificate (a trusted anchor, usually provided using some secure channel) to the signer's certificate needs to be checked. This includes the processing of revocation information and validity periods for all involved certificates. Given the mathematical correctness of all signatures in the path, the validity of a signature depends on the validity model used for the validation of the certification path. The validity model specifies how the revocation information and the validity periods are evaluated. In literature, three validity models can be found, which we shortly explain adhering to the definitions from [3].

To formally describe the models let N be the length of the certification path. The index $k = 1$ is assigned to the Root CA and $k = N$ is assigned to the end-entity of the chain, i.e. the creator of the document's signature to be verified. $Cer(k)$, the kth certificate in the chain, certifies the key of the kth participant, where the certificate of the Root CA in general is self-signed. According to [3] we denote with $T_i(k)$ the starting date of the validity period (normally the issuance date) of $Cer(k)$ and with $T_e(k)$ the expiry date. T_s denotes the time of signature generation by the end-entity and T_v the time of verification of a signature. Note that while the knowledge of T_v is trivial for the verifier, the knowledge of T_s is not and requires a trustworthy time information, i.e. a time-stamp by a trusted third party.

Shell Model

Definition 1 (Shell Model). *A digital signature is valid at verification time T_v if:*

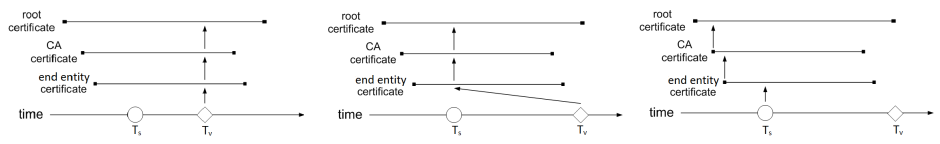
1. All certificates in the certification path are valid at T_v: $T_i(k) \leq T_v \leq T_e(k)$ for all $1 \leq k \leq N$ and no certificate is revoked at T_v.
(2. The end-entity certificate $Cer(N)$ is valid at signing time T_s: $T_i(N) \leq T_s \leq T_e(N)$ and it is not revoked at T_s.)

Figure 1a shows the shell model. For a successful verification all certificates in the chain (including the end-entity's certificate) have to be valid at the time of signature verification. Property 2, as found in [3], implies that all certificates additionally were valid at the time of signature generation. Yet, the PKIX standard RFC 5280 [7] for example does not consider property 2. In that case, T_s is completely irrelevant for the validity of a signature, which might be suitable in an SSL/TLS scenario as $T_s \approx T_v$. Currently most of the applications implement the shell model for certificate path validation [3].

Extended Shell Model

Definition 2 (Extended Shell Model). *A digital signature is valid at verification time T_v if all certificates in the certification path are valid at T_s: $T_i(k) \leq T_s \leq T_e(k)$ for all $1 \leq k \leq N$ and no certificate in the path is revoked at T_s.*

In the extended shell model (also hybrid or modified shell model) T_s is used instead of T_v during validation. That means, the certificates in the chain are checked for validity and revocation state at generation time of the end-entity signature. Figure 1b depicts the model. To implement this model, the signature generation time needs to be bound to the signature such that it can be checked that the certificates in the path were not revoked and were valid at that time.



(a) shell model (b) extended shell model (c) chain model

Fig. 1: Validity Models (with signature generation time T_s and verification time T_v, vertical arrows show the point in time used for validation of the superordinate certificate)

Chain Model

Definition 3 (Chain Model). *A digital signature is valid at verification time T_v if:*

1. *The end-entity certificate $Cer(N)$ is valid at the signing time T_s: $T_i(N) \leq T_s \leq T_e(N)$ and $Cer(N)$ is not revoked at T_s.*
2. *Every CA certificate in the chain is valid at the issuance time of the subordinate certificate in this chain: $T_i(k-1) \leq T_i(k) \leq T_e(k-1)$ and the certificate $Cer(k-1)$ is not revoked at $T_i(k)$ for all $2 \leq k \leq N$.*

In the chain model, any signature in the chain is validated using its signature generation time, i.e. the issuance time of the certificate. For simplicity, the start date of the validity period can be used as an approximation (see Figure 1c). Thus, this date must lie within the validity period of the superordinate certificate.

Using time-stamps for realization, a time-stamp is required for every signature within the chain [3]. This is because dates prior to the signature generation time of the end-entity signature have to be considered for the chain validation.

3 Problem statement

Considering the case of a CA key compromise and the subsequently required revocation of that key, the main problem is the implicit revocation of all certificates and signatures that rely on it. This includes all signatures created using the compromised key or any key certified directly or transitively using the compromised key. Therefore, there is a total breakdown of the infrastructure spanned by the compromised key. The implicit revocation in turn is required by the difficulty of distinguishing between genuine and forged signatures.

This could be avoided by multiple independent issuers certifying the same keys and thereby spanning independent, redundant PKIs. As this introduces a considerable overhead, the work at hand assumes that there is no additional certification path not containing the compromised keys.

While in the shell model distinguishing between genuine and forged signatures is not considered at all, in the other two presented models (cf. Section 2.2) it is addressed based on the signature generation times. However, the signature generation times must be provided in a secure way. Given this is realized somehow, we review the impact caused by the compromise of any CA within a PKI in conjunction with the applied model for signature validation. Electronic signatures are basically used for two purposes, signing electronic documents and authentication. We differentiate between these two cases.

Unsuitability of the Shell model. From a security point of view, the shell model must be used in case it is impossible to distinguish between legitimately and maliciously generated signatures after a key compromise. Using the shell model, all signatures depending on the compromised key are considered invalid. Note that the expiration of a certificate has the same effect as a revocation. Any signature and thus subordinate certificates become automatically invalid at some point in time due to either expiration or revocation of any certificate within the certification path. Therefore, in case any CA certificate is revoked or expires, all subordinate certificates in the chain have to be renewed. This might be manageable in case of scheduled expiration, as maintenance measures like issuing new certificates can be planned and executed beforehand. However, in case of a sudden compromise this leads to an immediate breakdown of large parts of the infrastructure in case a major CA is concerned. In other words, if a CA is compromised it is no longer possible to authenticate or validly sign documents until the certificates are renewed. Even worse, all signatures issued before the incident become invalid. Nevertheless, such a renewal is theoretically not necessary if it is possible to securely verify whether a signature was issued prior to the expiration or revocation of any certificate in the chain. Hence, the shell model is unsuitable because such a verification is not considered there.

Partial unsuitability of the Extended Shell model. If the extended shell model is used and is securely implemented using an appropriate dated proof of existence, signatures generated before a compromise and revocation stay valid. Yet, a compromise of a CA's private key still has the same effect considering authentication. This is due to the fact that the signature generation in this case is naturally related to the current time. Additionally, signatures on documents cannot be further validly generated either. Thus, although keeping former signatures valid, certificate renewal is necessary to the same extend as when using the shell model.

Suitability of the Chain model. The chain model considers a certificate trustworthy if it is not revoked nor expired and was issued before a potential superordinate CA key compromise or certificate expiration. Furthermore, it considers the validity of any certificate in the certification path at the time the respective key was used for the signature generation. As a consequence, subordinate certificates remain valid for signature creation and verification, and therewith also for authentication, even after the invalidation of superordinate certificates. Thus, the properties of the chain model minimize the impact of a CA compromise and completely resolve the unintended impact on dependent certificates. Nevertheless, the model requires a secure means to assure the chronological order between the events in a PKI. For instance, for the chain model to work it is required to know when a compromise occurred and which signatures were issued before and which ones after the compromise.

The chain model thus behaves as desired and is, for example, mandatory for the validation of qualified electronic signatures in Germany (cf. [13]). Nevertheless, the difficulty of achieving secure proof of existence has to be overcome.

The proof of existence problem. Time-stamps are the current solution to provide a proof of existence. Even though other proposals exist, these are rather impractical and barely used in practice (see Section 6). However, time-stamping is not an optimal and complete solution: First, time-stamping requires the setup and maintenance of an additional and independent TSA infrastructure, and the trustworthiness of the TSAs to apply the correct date and time. Second, time-stamps rely on electronic signatures themselves, thus face the same problems concerning compromise and expiration as common electronic signatures do. That is, upon the compromise of a TSA or any superordinate CA, the issued time-stamps become invalid and the proof of existence is lost. Therefore, time-stamps only defer the problem to the TSA infrastructure. Third, time-stamps increase the overhead of digital signature validation, since they are signed data objects whose validation is also mandatory. And fourth, an online action is required to obtain the time-stamp. This slows down the signing procedure.

4 Solution

We show how to solve the proof of existence problem by using a forward secure signature scheme (FSS), thereby we minimize the impact of a CA key compromise. First we introduce the concept of FSS. Then we show how to implement the chain model without the need for a TSA and time-stamping using an FSS.

4.1 Forward Secure Digital Signature Schemes

The idea of forward security for digital signature schemes was introduced by Anderson [2] and later formalized in [4]. In one sentence, the forward security property says that even after a key compromise, all signatures created before remain valid. Now we describe this in more detail. In currently known FSS, the lifetime of a key pair is split into several time periods, say T. We assume T is specified within the certificate. These time periods can be defined in different ways. Either it is done in terms of time, i.e. one time period corresponds to one day. Or the number of created signatures can be used, i.e. a time period ends after the key was used to create a certain number of signatures. Yet, in general anything that seems suitable can be applied to define the time periods. Note that this implies that the length of two time periods might differ. Especially, it is possible to associate the time periods with single signatures. As in the case of a traditional digital signature scheme, an FSS key pair has one public key that is valid for the whole lifetime of the key pair. But, in contrast to a traditional signature scheme, an FSS key pair has many secret keys $sk_1, \ldots, sk_T$; one secret key for each of the T time periods.

The key generation algorithm of an FSS takes T as an additional input. To sign a message, the current secret key is used and the produced signature contains the index of the current time period. A signature is verified as valid if the signature is a valid signature on the message under the given public key and the index of the time period included in the signature. The private key is updated for the following time period using an additional key update algorithm. This algorithm is either called manually by the user, scheduled to run at the end of the time period, or is part of the signature algorithm, depending on the way the time periods are defined.

A FSS provides the same security guarantees as a traditional signature scheme. Even if an adversary can trick the user into signing messages of her choice, this adversary is unable to forge signatures on subsequent messages. But the forward security property also gives a stronger security guarantee: Even if the adversary learns the current secret key sk_i of time period i, then the adversary is unable to forge a signature for any time period $j < i$. For a formal definition of FSS we refer the reader to [4].

4.2 FSS and Revocation

The forward security property allows us to handle revocation in a fine grained manner. Revocation is a means to limit abuse of a certificate. A certificate is revoked in case of a key compromise or for organizational reasons. In the following we only look at the case of key compromise, the other cases follow accordingly. In case of a key compromise, the forward security property guarantees that all signatures created prior to the compromise originate from the certificate owner. So there is no need to render these signatures invalid. As all signatures contain the index of the time period they were created, we can do a fine grained revocation. We do not revoke the validity of the certificate in general, but we revoke only the validity for all time periods starting from the time period when the key was compromised. So if we know the index c of the time period of key compromise, the revocation starts at index c. A signature including index i is accepted as valid if $i < c$ and invalid if $i \geq c$.

4.3 Chain model with FSS

Now we show how to realize the chain model, taking advantage of the fine grained revocation introduced above. First, we show how the traditional chain model can be securely realized without a TSA. This realization takes validity periods for certificates into account, which requires the binding of signature generation time and real time in form of a calendar date. Then, we argue from a security point of view, that there is no reason for explicit validity periods when using FSS. We discuss the possibility to abandon explicit validity periods based on calendar dates and show how to handle implicit validity periods given by the indices. Thus, realizing the chain model only working on indices. This is especially relevant for FSS where the time periods for key update are not defined using calendar dates, but the number of generated signatures. While both approaches consider the exclusive usage of FSS, we show in the third scenario that the chain model can be securely implemented without a TSA in the context of authentication, even in case the end-entity does not use FSS.

Realization with validity periods. Currently, the definition of the chain model is tailored to the binding of signature generation time and real time in form of a calendar date. Binding a time in form of a calendar date to the signatures can easily be realized for FSS by simply including the signature generation time into the signature and signing it together with the signed data. By the forward security, such a time-tag cannot be forged by an adversary that later compromises the key. The point is, that an adversary can only use later key states to sign, thus even back dated time-tags become invalidated by a revocation. Therefore, such time-tags are secure. The trustworthiness of CAs is a preliminary for the concept of hierarchical PKIs thus time tags included by CAs are trustworthy by assumption. End-entities might have appeals to forge such time-tags. Nevertheless, this is another problem not considered it in this work.

To implement the chain model with FSS, we propose with Definition 4 a slightly adapted version of Definition 3 as seen in Section 2.2. The variables are defined as above. Signature generation times can be extracted from self-signed time-tags. We furthermore assume an FSS, where the time period is represented by a running index and that the end-entity uses FSS, too. Additionally, let I_s be the index used for end-entity signature generation, $I_s(k)$ the signing index used to sign certificate k and $I_r(k)$ a possible revocation index. Additionally, $I_e(k)$ denotes the maximal index specified in certificate k. Thus, $I_r(k) = I_e(k) + 1$ in case there is no revocation for certificate k. Note that due to this definition $I_s(k)$ and $I_r(k-1)$ are indices belonging to the same key pair.

Definition 4 (Chain Model with FSS v1). *A digital signature is valid at verification time T_v if:*

1. *The end-entity certificate $Cer(N)$ is valid at the signing time T_s: $T_i(N) \leq T_s \leq T_e(N)$ and I_s is not revoked for $Cer(N) : I_s < I_r(N)$.*
2. *Every CA certificate in the chain is valid at the issuance time of the subordinate certificate in this chain: $T_i(k-1) \leq T_i(k) \leq T_e(k-1)$ and $I_s(k)$ is not revoked for certificate $Cer(k-1) : I_s(k) < I_r(k-1)$ for all $2 \leq k \leq N$.*

As CAs in general sign certificates, the time-tag can be included into the certificate itself. To have an exact date, this could be an additional field called `IssuanceDate`, which is then signed as a part of the certificate. This would enable the issuance of certificates that become valid at a later point in time i.e. $IssuanceDate < NotBefore$. An approximate solution, which does not require a new field, would be to use the `NotBefore` date already included in X.509 certificates. This is less precise but an acceptable approximation for the issuance date in most cases. In this case it has to be considered that the validity of issued certificates must start within the validity period of the issuer's certificate in order to be positively validated.

Realization without validity periods. Here we consider a chain model only based on the chronological ordering of signatures and without explicit calendar dates. Getting completely rid of calendar dates makes the implementation more efficient as their verification is omitted. Furthermore, abandoning calendar dates implies that time synchronization is non-essential for the security.

The chain model without explicit calendar dates is given in Definition 5. Revocation is based on the index and does not depend on the exact calendar date when a signature was generated. Implicit validity periods are ensured by validating the signature index. The variables are as defined in Section 4.3.

Definition 5 (Chain Model with FSS v2). *A digital signature is valid at verification time T_v if:*

1. *The end-entity certificate $Cer(N)$ is valid for signing index I_s: $1 \leq I_s \leq I_e(N)$ and I_s is not revoked for $Cer(N) : I_s < I_r(N)$.*
2. *Every CA certificate in the chain is valid for the signing index used for the subordinate certificate in this chain: $1 \leq I_s(k) \leq I_e(k-1)$ and $I_s(k)$ is not revoked for $Cer(k-1) : I_s(k) < I_r(k-1)$ for all $2 \leq k \leq N$.*

The rationale behind explicit validity periods based on calendar dates is twofold. On the one hand, the business model of CAs is based on the recurring issuance of certificates. On the other hand it has security reasons, as validity periods enforce a key renewal after a certain time helping to address a possible fading out of algorithms.

In case of FSS, it is reasonable to abandon explicit validity periods, as such schemes in general only allow for a certain number of signatures or a validity period is implicitly given by the construction of the scheme. Thus, for example, an FSS certificate becomes valid at the time of issuance and the validity period ends when a given number of indices specified within the certificate is used up.

Nevertheless, in order to ensure that a certificate is only used for a fixed time period, e.g. 2 years, there are several possibilities. The first would be to split the number of allowed indices evenly over the desired validity period and publicly announce which indices have to be used on which day[3]. The certificate owner must then increase the index, e.g. on a daily basis, to adhere to the schedule. Yet, this approach has several disadvantages. The certificate owner can only generate a limited number of signatures in any sub period. On the other hand, if not using up all indices, he must manually update the index and the key.

The second possibility would be to revoke a certificate at the end of the desired validity period with the current index, thus not affecting signatures generated before. This has the advantage that no fix schedule and no manual increase of the index are needed. The only requirement is the proper revocation, which requires the key owner to correctly report the current index to the CA.

The third possibility to achieve an approximate validity period would be to individually estimate the number of signatures to be generated within the validity period and set the final index according to that estimation. This might lead to a somewhat shorter or larger validity period as desired depending on the difference between the estimated and the actual number of generated signatures.

Realization with stateless (non FSS) end-entity signatures. In the case of authentication or similar use cases, where the signature time T_s is approximately the verification time T_v, one does not gain any benefit from using FSS

[3] This would also solve the problem of users putting fake time-tags to some degree.

for the end-entity certificate. As FSS are slightly less efficient then traditional signature schemes and as they are stateful, this might even be a drawback in some scenarios. Therefore we show that even if the end-entity uses a conventional (stateless) signature scheme, the chain model can securely be realized without a TSA in case the signature time T_s is approximately the verification time T_v. For instance, in the SSL/TLS Internet setting. Nevertheless, a TSA is indispensable if any CA does not use FSS. Definition 6 shows the special case where T_v replaces T_s in step one. Note that the revocation of end-entity certificates must be done in the conventional way.

Definition 6 (Chain Model with FSS v3). *Given all signature schemes involved in the certification path are FSSs except the end-entity scheme and $T_s \approx T_v$, then a digital signature is valid at verification time T_v if:*

1. *The end-entity certificate $Cer(N)$ is valid at the verification time T_v: $T_i(N) \leq T_v \leq T_e(N)$ and $Cer(N)$ is not revoked at T_v.*
2. *According to Definition 4 if using explicit calendar based time period, or Definition 5 otherwise.*

Even in case the end-entity uses a common stateless signature scheme for document signatures, where $T_s \approx T_v$ does not hold in general, FSS for CAs eases the use of the chain model. A TSA would then only be needed to time-stamp the end-entity signature. This time-stamp has to be validated during signature verification. No further time-stamps concerning the certificates are necessary, thus the chain model can be implemented as described in Definition 6 replacing in step 1 T_v by T_s as is contained in the time-stamp.

5 Implementation

In this section we discuss how to implement our proposal. First we show how to change the path validation specified in RFC 5280 [7]. Then we discuss some revocation related issues, including how to determine the time period of a key compromise. We show how to improve the revocation handling on the client side and we discuss which FSS fits to our proposal.

5.1 Implementation of the chain model

We suggest the usage of FSS for CAs to implement the chain model for path validation in a practicable way (see Section 4.3). The implementation only requires small changes in the path validation specified in RFC 5280 [7]. The changes are shown in Algorithm 1. The new parameter *working_validity_period* and the italic parts are only needed if explicit validity periods based on calendar dates in FSS certificates are considered. Otherwise the signature generation time is considered to be implicitly given by the used signature index. If the used index is smaller than a potential revocation index, then the certificate has been validly generated. In case of the end-entity certificate (i.e. step c), we assume a standard signature scheme. In that step shell and chain model are equivalent given the current time is approximately the signing time.

Algorithm 1 Basic Certificate Processing Chain model with FSS

```
Initialize working_public_key_algorithm, working_public_key, working_public_key_parameters, and
working_issuer_name *(and working_validity_period)* with the values from the trust anchor information.
For each certificate in the certificate chain do:

 1. Basic Verification
    (a) Check if signature_index contained in the certificate signature is revoked for
        working_public_key (and if issuance date of certificate is within working_validity_period).
    (b) Verify certificate signature using signature_index, working_public_key_algorithm, the
        working_public_key, and the working_public_key_parameters.
    (c) if current is end-entity certificate: Check if the certificate validity period includes the
        current time and is not revoked at the current time.
    (d) Check if the certificate issuer name is the working_issuer_name.
 2. as described in [7]

Prepare for next certificate in the chain.
Wrap-Up Procedure
```

The variable *signature_index* is contained in the signatures generated with an FSS and can be extracted from the processed certificates. Using that index for revocation as described in Algorithm 1, one major difference to RFC 5280 [7] is that the revocation information checking for the root certificate is enforced. We consider this to be an essential security aspect in case of potential CA compromises. Currently, all browsers come with a built in list [32] of trusted root certificates. The self-signed root certificates are allowed to be excluded from path validation (see e.g. TLS 1.0 RFC 2246 [8]) and only be used for initialization. Thus, no revocation information is checked for these trusted Root CA certificates and any revocation in case of a compromise remains without effect. On the one hand, this is a design issue as in the traditional setting CAs use the same key for certificate and CRL signing which is insecure [25] if Root CAs revoke their own key. Yet, there are two other settings backed by the X.509 standard, namely that different keys are used or revocation is performed by another entity. In these settings revocation of Root CA keys is possible.

Furthermore, the client applications (e.g. web browsers) have to support the used FSS schemes. To obtain a broad applicability, the verification algorithms of the FSS of choice as well as the path validation according to the chain model have to be implemented within major libraries and crypto providers, as e.g. OpenSSL [27].

5.2 Compromise Detection and Revocation

In Section 4.2 we discuss how to realize fine grained revocation using an FSS. This requires finding out the key index during the key compromise. Commonly, this is done as follows. First of all, the certificate holder has to determine the date of key compromise. How this can be realized depends on the kind of key compromise, but is not addressed in this work. One possible approach is described in [33]. Knowing the compromise date, it needs to be linked to an FSS time period and the corresponding index. Now, there are several ways to define the FSS time periods (see Section 4.1). If the FSS time periods are defined in terms of real time, this directly gives us the index of the FSS time period at key compromise and we are done. Nevertheless, it is also possible that the FSS time periods are defined in another way, such as one signature per time period. In this case, it is not easily possible to link a date to an index. Therefore, the certificate owner has to keep track of his key updates. In case of a CA, this can be done logging the last used index with the current date to a write once memory during key update. Then, given the date of the key compromise, it is possible to determine the last index used before the compromise even if the adversary manages to suppress the logging. Alternatively, it would be possible to publish the last used index at the end of each day in a newspaper.

5.3 Further advantages and choice of FSS

An additional advantage of FSS is that their statefulness allows to improve the client-side revocation handling by locally maintaining the states of the known CA certificates by storing the highest index currently known to be valid. In contrary to current practice [29], this allows the enforcement of so-called hard fails at least for CA certificates while minimizing the impact of unavailability issues. Furthermore, due to the precise impact of revocation in our approach, a certificate can be revoked on suspicion of a compromise.

Although our proposal works with arbitrary FSS, we propose to apply the eXtended Merkle Signature Scheme (XMSS) [5] as it is most suitable for the given scenario. XMSS is hash-based and thus a post quantum signature scheme. It is as fast as RSA and ECDSA although it is forward secure and it provides comparable key sizes. For XMSS, a time period is hard linked to one single signature and the key is automatically updated by the signature algorithm. In case of a CA, this allows the most fine grained revocation handling that is possible.

Furthermore, with XMSS the probability of a sudden breakdown caused by advances in cryptanalysis can be efficiently minimized [5]. On the one hand, it requires minimal security properties to be secure, thus the break of harder properties can be seen as an early-warning system. On the other hand, so-called hash combiners (i.e. see [11]) can be used, such that the resulting combination is secure as long as at least one of the hash function families is secure. For more details on the several advantages see the extended version.

6 Related Work & Comparison

Key compromise and revocation can cause a huge impact on PKI systems, which is a well known problem. Researchers (e.g. [16]) have criticized how revocation is implemented in X.509. In this context, several proposals came up to either avoid revocation or mitigate its impact as shown in the following.

The complete elimination of revocation in PKIs by the use of short lived certificates is proposed by Rivest [28] and applied by e.g. Gassko et al.[12]. Yet, this approach comes with a considerable overhead of repeated certificate issuance and, in case of CA certificates, rebuilding the whole certificate hierarchy.

Other authors propose to distribute trust among multiple instances. While Maniatis et al. [14] propose a Byzantine-fault-tolerant network of TSAs to provide protection against TSA compromise, Tzvetkov [31] proposes a disaster coverable PKI model based on the majority trust principle. The first uses additional proofs of existence based on threshold signatures but requires a complex infrastructure and generates a huge overhead during verification of the signatures and time-stamps. In the latter, to tolerate the compromise of a minority of CAs, each certificate has to be signed in parallel by different CAs.

The use of write-once and widely witnessed media (e.g. official gazettes or newspapers) is an alternative to anchor digital objects in the time-line. Combined with the application of hash chains, as done by the TSA Surety [30], this can be implemented more efficient, but the usability and the preservation (e.g. of printed journals) in long-term raise concerns.

Baier and Karatsiolis [3] identify and define the three different validity models. They also propose an implementation of the extended shell and the chain model based on time-stamps using the ETSI specification CAdES (CMS Advanced Electronic Signatures [10]) to provide the required proofs of existence. Yet, this leads to quite complex realizations due to the additional management of the time-stamps and their verification during path validation.

Although the use of FSS in the area of PKI is not novel, no previous work has yet provided a complete forward secure PKI model to the best of our knowledge. Besides our work, Kim et al. [20] propose to use FSS for CAs to ensure business continuity in case of a CA key compromise. Thus, they use FSS in the same way as in our approach. Yet, their work lacks a model for path validation in hierarchical PKIs, which is highly interwoven with the properties of FSS and must be adequately chosen to exploit the specific advantages. We show that in order to achieve business continuity, i.e. validly generated certificates stay valid even in case of a CA key compromise, the chain model must be used and that the common shell model is not suitable. Besides that, Kim et al. [20] consider FSS with fixed key update periods. Our approach is more general. To the best of our knowledge we are the first showing how revocation must be handled to obtain fine grained revocation. Additionally, we show how to omit explicit validity periods, simplifying path validation without a security decrease.

Several other works apply FSS within PKI [15,21,33,23,24], but there are significant differences in the goals and the use of FSS compared to our work. Go [15] considers FSS for CAs, yet within the threshold setting in mobile AD Hoc networks – which significantly differs from our PKI setting – concluding that no existing scheme fulfills the specific requirements.

Koga et al. [21] propose a PKI model where the certificate chain for validation always has length one. They propose different constructions where either FSS or key-insulated signatures schemes (KIS) are used by the Root CA to generate the secret keys for the Sub CAs. While this allows the keys of Sub CAs to stay valid in case of a Root CA compromise, in case of the construction based on FSS, the compromise of a Sub CA implies the compromise of Sub CAs that obtained keys with higher indices. Multiple Root CA key pairs or KIS solve this problem but at the cost of additional overhead. The KIS approach is further developed by Le et al. [23]. Nevertheless, in both works [21,23] CAs always use their unique key to sign common user certificates. Thus, all user certificates issued by a certain CA are invalidated in case of this CA being compromised. Furthermore, the CA keys need to be securely transferred from the Root to the Sub CAs. The approach of Le et al. even needs the transport of tamper resistant sub devices to the CAs.

In another paper Le et al. [24] propose to use FSS in revert order to allow to easily invalidate signed credentials. That is, a credential is signed with many keys obtained from an FSS key pair. Credentials can be invalidated by successively publishing the keys in reverse order. While this could be used to obtain short lived certificates, the applicability to establish a PKI is limited. As a key pair can obviously only be used to sign a single document this would imply the management of a huge amount of signing keys that are exposed to a possible compromise as the reverse order does not allow the deletion of former FSS keys.

While our focus lies on the PKI and its mechanisms for revocation and validation itself, Xu and Young [33] consider compromise detection to finally obtain a robust system. To keep track of the key usage, signatures are deposited at highly secured systems and published on bulletin boards, where FSS are used to provide a stateful authentication. That means tampering can be observed based on inconsistencies in the authentication key states.

7 Conclusion

We saw that in the shell model a CA revocation results in a complete transitive invalidity of all dependent certificates and signatures, leading to a total break down of the respective PKI. We argued how the alternative validity models, i.e. extended shell model and chain model, improve the situation and demonstrated their dependency on dated proofs of existence. We showed that time-stamping, the current way to provide these proofs, inflates the infrastructure and increases the overhead for signature generation and validation while facing the same problems it is supposed to solve. We proposed different ways to utilize FSS to create certificate inherent dated proofs of existence, thereby getting rid of time-stamps and their infrastructure. We saw that the use of FSS is an attractive solution regarding revocation and to mitigate its impact compared to alternative mechanisms. First, because it does not add extra costs for the maintenance of a PKI, e.g. caused by key and certificate renewal after short periods to prevent revocation. Second, from the end-user's point of view, the probability that his certificate becomes invalid because of a CA key compromise is minimized without any costs. The usability and the computational effort to use the PKI services is equal to conventional signature schemes. Finally, our solution adds no new constraints. By giving advice on implementation issues, we showed how our solution also supports more aggressive revocation policies, simplifies client side revocation checking, and enables sudden breakdown protection. All in all, we showed how to substantially mitigate the probability and the impact of CA certificate revocation and at the same time get rid of time-stamping infrastructures.

Acknowledgements

Parts of this work were funded by ISPRAT e.V., Hongkongstraße 3, 20457 Hamburg, Germany. The second author was supported by grant no. BU 630/19-1 of the German Research Foundation (`www.dfg.de`).

References

1. C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161 (Proposed Standard), Aug. 2001. Updated by RFC 5816.
2. R. Anderson. Two remarks on public key cryptology. In *Manuscript. Relevant material presented by the author in an invited lecture at the 4th ACM Conference on Computer and Communications Security, CCS*, pages 1–4. Citeseer, 1997.
3. H. Baier and V. Karatsiolis. Validity models of electronic signatures and their enforcement in practice. In *Proceedings of the 6th European conference on Public key infrastructures, services and applications*, EuroPKI'09, pages 255–270, Berlin, Heidelberg, 2010. Springer-Verlag.
4. M. Bellare and S. Miner. A forward-secure digital signature scheme. In M. Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, volume 1666 of *LNCS*, pages 786–786. Springer Berlin / Heidelberg, 1999.
5. J. Buchmann, E. Dahmen, and A. Hülsing. XMSS - a practical forward secure signature scheme based on minimal security assumptions. Cryptology ePrint Archive, Report 2011/484, 2011. `http://eprint.iacr.org/`.
6. Comodo. The Recent RA Compromise. `http://blogs.comodo.com/it-security/data-security/the-recent-ra-compromise/`, visited Nov. 2011.
7. D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Rfc 5280: Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280 (Proposed Standard), 2008.
8. T. Dierks and C. Allen. Rfc 2246: The tls protocol version 1.0, 1999.
9. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.
10. ETSI. Electronic signatures and infrastructures (esi); cms advanced electronic signatures (cades), 2008. TS 101 733 V1.7.4.
11. M. Fischlin, A. Lehmann, and K. Pietrzak. Robust multi-property combiners for hash functions revisited. In L. Aceto, I. Damgård, L. Goldberg, M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *Automata, Languages and Programming*, volume 5126 of *Lecture Notes in Computer Science*, pages 655–666. Springer Berlin / Heidelberg, 2008.
12. I. Gassko, P. Gemmell, and P. MacKenzie. Efficient and fresh certification. In *Public Key Cryptography*, pages 342–353. Springer, 2000.
13. German Federal Network Agency. Faq (frequently asked questions). `http://www.bundesnetzagentur.de/cln_1911/DE/Sachgebiete/QES/FAQ/faq_node.html`. accessed December 2011.
14. P. Giuli, P. Maniatis, T. Giuli, and M. Baker. Enabling the Long-Term Archival of Signed Documents through Time Stamping. In *Proceedings of the 1st USENIX Conference on File and Storage Technologies*, FAST '02, pages 31–46. USENIX Association, 2002.

15. H.-w. Go. Forward security and certificate management in mobile ad hoc networks. Master thesis, University of Hong Kong (Pokfulam Road, Hong Kong), 2004.
16. P. Gutman. PKI: It's Not Dead, Just Resting. *Computer*, 35:41–49, 2002.
17. h online. Attack on Israeli Certificate Authority. `http://h-online.com/-1264008`, visited Nov. 2011.
18. h online. Fake Google certificate is the result of a hack. `http://h-online.com/-1333728`, visited Nov. 2011.
19. ISO. *ISO/IEC 9594-8 - Information technology — Open Systems Interconnection — The Directory: Public key and attribute certificate frameworks*. International Organization for Standardization, 2008.
20. B. Kim, K. Choi, and D. Lee. Disaster coverable pki model utilizing the existing pki structure. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, volume 4277 of *LNCS*, pages 537–545. Springer Berlin / Heidelberg, 2006.
21. S. Koga and K. Sakurai. Decentralization methods of certification authority using the digital signature scheme. In *2nd Annual PKI Research Workshop – Pre-Proceedings*, pages 54–64, 2003.
22. L. Kohnfelder. *Towards a practical public-key cryptosystem*. PhD thesis, Massachusetts Institute of Technology, 1978.
23. Z. Le, Y. Ouyang, J. Ford, and F. Makedon. A hierarchical key-insulated signature scheme in the ca trust model. In K. Zhang and Y. Zheng, editors, *Information Security*, volume 3225 of *Lecture Notes in Computer Science*, pages 280–291. Springer Berlin / Heidelberg, 2004.
24. Z. Le, Y. Ouyang, Y. Xu, J. Ford, and F. Makedon. Preventing unofficial information propagation. In S. Qing, H. Imai, and G. Wang, editors, *Information and Communications Security*, volume 4861 of *Lecture Notes in Computer Science*, pages 113–125. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-77048-0_9.
25. M. B. MBarka and J. P. Stern. Observations on certification authority key compromise. In *EuroPKI*, pages 178–192, 2010.
26. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 2560 (Proposed Standard), June 1999. Updated by RFC 6277.
27. OpenSSL. Cryptography and SSL/TLS toolkit, 2012. `www.openssl.org/`.
28. R. L. Rivest. Can We Eliminate Certificate Revocations Lists? In *FC '98: Proceedings of the Second International Conference on Financial Cryptography*, pages 178–183, London, UK, 1998.
29. spiderlabs. Defective By Design? - Certificate Revocation Behavior In Modern Browsers, visited Dec. 2011. `http://blog.spiderlabs.com/2011/04/certificate-\\revocation-behavior-in-modern-browsers.html`.
30. Surety. Protecting Intellectual Property with a Digital Time Stamp, 2011. `http://www.surety.com/`.
31. V. Tzvetkov. Disaster coverable pki model based on majority trust principle. *Information Technology: Coding and Computing, International Conference on*, 2:118, 2004.
32. N. Vratonjic, J. Freudiger, V. Bindschaedler, and J.-P. Hubaux. The Inconvenient Truth about Web Certificates. In *The Workshop on Economics of Information Security (WEIS)*, 2011.
33. S. Xu and M. Yung. Expecting the unexpected: Towards robust credential infrastructure. In R. Dingledine and P. Golle, editors, *Financial Cryptography*, volume 5628 of *Lecture Notes in Computer Science*, pages 201–221. Springer, 2009.