Revocation & Non-Repudiation: When the first destroys the latter

Johannes Braun[1], Franziskus Kiefer[2], Andreas Hülsing[1]

[1] TU Darmstadt, Germany {jbraun|huelsing}@cdc.informatik.tu-darmstadt.de
[2] University of Surrey, United Kingdom f.kiefer@surrey.ac.uk

Abstract. Electronic signatures replace handwritten signatures in electronic processes. In this context, non-repudiation is one of the most desired properties – yet in practice it cannot be provided by the signature schemes themselves. Therefore, additional mechanisms in the underlying public key infrastructure are required. In this work, we present a formal treatment of that issue. We extend the formal model for public key infrastructures by Maurer introducing transitions to make it dynamic. We use the extended model to evaluate the relationship between non-repudiation and revocation and prove that backdated revocation always destroys the non-repudiation property. We prove that forward secure signatures can be used to maintain non-repudiation, rendering the costly use of time-stamping – as required by all existing solutions – superfluous. We also show how to realize this in practice, introducing a new index reporting protocol. Moreover, we show how this protocol can be used to support detection of malicious key usage, thereby improving the overall security of electronic signing. Besides, the index reporting protocol allows for a convenient realization of pay per use models for certificate pricing.

1 Introduction

Over the past few years, the importance of eBusiness and eGovernment has been steadily growing. More and more processes are handled online without physical interaction. To guarantee for authenticity and non-repudiation in such processes, electronic signatures are used. Moreover, many countries allow to replace handwritten signatures by electronic signatures and consider these as legally binding [17]. This allows to transfer many processes to the digital world that formerly required a media disruption, e.g. in many countries applying for a bank account. However, there is a fundamental difference between handwritten and electronic signatures. While handwritten signatures are naturally bound to a single person, the binding between electronic signatures and a person is artificial and thus fragile. The private key, required to generate signatures, can be applied by anyone who knows it or has access to it, without any possibility to distinguish which signature has been generated by whom. Thus, electronic signatures can only provide authenticity and non-repudiation as long as the private signature key is only applicable by a single person. While for authentication exclusive applicability during signature generation is only checked once, for use-cases that require

non-repudiation it must be provable as long as the signature is of any interest. In many cases, non-repudiation must be preserved for ten years and more by law. So far, this problem was never approached using a formal analysis. We make up for this omission. Namely, we present a formal treatment of the problem of preserving non-repudiation in practice. Additionally, we show how this problem can be solved more efficiently than today.

The binding between a specific key and a person is realized by a (hierarchical) Public Key Infrastructure (PKI). In a PKI the binding between the signer's identity (e.g. a name) and his public key is established using certificates, issued (i.e. signed) by a Certification Authority (CA). This CA is also responsible for a revocation of the certificate in case of a key compromise. This is necessary, as there does not exist any usable and perfect protection of the secret key. However, it is necessary to protect the secret key in a way that a key compromise can be detected and that the key is protected at least until it is revoked. For this reason secure storage devices like smart cards should be used. In case of legally binding signatures a secure storage device is even required by law in most countries. But even secure storage devices can not guarantee perfect security. The device can be stolen or get lost. While these devices can be assumed to protect the secret key for short time span they should not be assumed to protect the key for a long time period, i.e. years, if an adversary has direct access. The progress in cryptanalysis, especially in the field of side-channel attacks, periodically proves such assumptions wrong. Anyhow, as revocation exists, the secure storage device only has to protect the secret key until the key compromise is detected and the key is revoked.

To summarize, the binding between key and a person is only temporary, terminated either by expiry of the certificate or revocation. And this is where the non-repudiation property, which is guaranteed by the electronic signature in theory fails in reality if there are no additional measures. As compromise is possible, a key owner can simply claim that his key was compromised, ascribing the generation of signatures to an adversary and thereby repudiating valid signatures.

To prevent such a repudiation attack, a chronological order of events is required and must be considered during signature validation. A signature should then be verified as valid, if it was generated before a key compromise. In practice, the actual result of a signature validation does not only depend on the verification result of the candidate signature but also on the validation model. Validation models for hierarchical PKIs define, which certificates in the certificate chain of the candidate signature have to be valid at which time for a successful validation. The current Internet standard for certificate validation, namely the shell model [9], cannot be used for legally binding signatures, as it does not take the order of events into account [3,7] and hence a signature becomes invalid if any certificate in the chain is expired or revoked. The chain model (cf. Section 2.1) in contrast is applicable as it takes into account the chronological order of events and allows to verify a signature as valid, if all signatures in the certificate chain and on the document were generated before the corresponding certificate was

revoked or expired. The crux of implementing the Chain model is to establish this provable chronological order of events.

Common signature schemes do not provide inherent properties to determine and prove the chronological order of generated signatures. Today's solution is to apply an indirection and use time-stamps generated by trusted third parties – an inefficient approach with plenty of disadvantages (see Appendix A). We propose a new solution using forward secure signature schemes (FSS) which have chronological ordering as an inherent property. But some challenges remain, namely how to establish a before and after relation between signature generation and revocation in the face of dishonest end-entities aiming at repudiating their own signatures. A similar approach was presented in [18], yet, problems like compromise detection and key update scheduling remain open. Furthermore, non-repudiation is established at the end of a time period and not directly when a verifier obtains a signature.

Contribution. Based on Maurer's formal model for PKI [15] and its extensions [14,5], we establish a generalized and dynamic PKI model, in which a before and after relationship between two events can be described. This allows us to describe non-repudiation, which was not possible in previous models. We show, that non-repudiation strictly requires the prevention of backdated revocation, proving equivalence of the two. We evaluate the problems with backdated revocation and discuss different methods to prevent it. We evaluate our model on the approach of trusted time-stamps in Appendix A.

We present a new solution to establish a provable chronological order of events based on FSS, called Sign & Report, and prove that it can be used to achieve non-repudiation. Besides that we show how the proposed solution allows for a complementary security mechanism to detect key compromises. We also present a practical realization of the Sign & Report approach. Its core is an index reporting protocol, which allows monitoring of key usage by a TTP. Compared to the current approach of trusted time-stamps, with our approach we save one online step, do not need an independent infrastructure and save the signature generation and validation of the time-stamps. Furthermore, this allows for pay per use signatures besides the targeted core functionalities.

2 Security Model

In this section we propose a new extension to the formal security model introduced by Maurer in [15]. This extension allows to model revocation and formally define the notion of non-repudiation by using transitions between states. In this work we are only concerned with malicious end-entities. For the sake of simplicity, we thus do not consider attacks against CAs. One might for example use [7] to handle these. Hence we assume CAs to be trustworthy and non compromised.

2.1 Formal PKI Model

In [15] Maurer introduced a formal security model for public key infrastructures later extended by Marchesini et al. [14] and Bicakci et al. [5]. We build our

model upon [14] as they introduce a smooth notion of how to handle time. We generalize their model in the sense that we do not depend on real time, but allow any indexing that admits a chronological ordering. This still includes the usage of real time information for indexing. While all former models are static, meaning they model one snapshot of a PKI, we introduce transitions between snapshots of the PKI, making the model dynamic. We further extend the model of [14], adding an explicit definition of revocation handling and end-entity signatures. This allows us to discuss non-repudiation using our model.

For simplicity, we only model relations starting from Sub-CAs that sign end-entity certificates. Thus, we assume trust in these Sub-CAs without considering how this trust is established. As we are only concerned with malicious end-entities, this fulfills our needs. It is straight forward to extend our model and proofs to the case of a hierarchical PKI. We also drop those parts of former models used to model a web of trust.

We model a PKI as View of a potential user at a specific time t. A user's View is a set of statements. We define six different statements. Trust expresses the trust in a (Sub-)CA, obtained according to the higher hierarchy or by explicitly trusting this CA. Cert says that the user has seen an end-entity certificate of the respective person. If a user has seen a certificate once, it remains in her view. The same holds for $\mathsf{Signature}$ and Revoc, which model that a user has seen a document signature or revocation information, respectively. Furthermore, there are two different Valid statements, which model that a user is convinced of the validity of an end-entity's certificate $C_{\alpha,\beta,\gamma,\varepsilon}$ or document signature $S_{\zeta,\eta,\delta}$. These two Valid statements can be inferred from other statements, using inference rules defined later. As we allow transitions between Views, every View is indexed with a time $t \in \mathbb{N}$. Note that indices used inside statements might be independent from the indices of the views. We write View^t for the View at time t and View if no specific t is needed.

Definition 1 (Statements). *Let* CA *denote a (Sub-)CA, $\mathcal{A}$ an end-entity's identity, D a document and $\mathcal{I}$ a (time) interval. A* $\mathsf{View}^t = \{\mathsf{stmt}_1, \ldots, \mathsf{stmt}_n\}$ *at point in time t consists of $n \in \mathbb{N}$ statements* stmt_i*. There exist the following six statements:*

$\mathsf{Trust}(\mathrm{CA}, \mathcal{I})$ *denotes the belief that, during the interval $\mathcal{I}$,* CA *is trustworthy for issuing certificates, i.e. models the axiomatic trust in (Sub-) CAs.*

$\mathsf{Cert}(\mathrm{CA}, \mathcal{A}, i, \mathcal{I})$ *denotes the fact that* CA *has issued a certificate for $\mathcal{A}$ at index i, which, during $\mathcal{I}$, binds $\mathcal{A}$'s public key to the certificate.*

$\mathsf{Signature}(\mathcal{A}, D, i)$ *denotes the fact that $\mathcal{A}$ has signed a document D at index i.*

$\mathsf{Revoc}(\mathrm{CA}, C_{\alpha,\beta,\gamma,\varepsilon}, i)$ *denotes the fact that* CA *has revoked the certificate $C_{\alpha,\beta,\gamma,\varepsilon}$, represented by statement $\mathsf{Cert}(\alpha, \beta, \gamma, \varepsilon)$, at index i.*

$\mathsf{Valid}(C_{\alpha,\beta,\gamma,\varepsilon}, i)$ *denotes the belief that certificate $C_{\alpha,\beta,\gamma,\varepsilon}$ is valid at evaluation index i.*

$\mathsf{Valid}(S_{\zeta,\eta,\delta})$ *denotes the belief that signature $S_{\zeta,\eta,\delta}$, represented by statement $\mathsf{Signature}(\zeta, \eta, \delta)$, is valid.*

A statement is `valid` if and only if it is in the View or can be derived from it using one of the inference rules defined below.

Signature Validation. We will now define the inference rules we use to validate signatures, i.e. derive `valid` for a Signature. The rules depend on the used certification path validation model. As mentioned in the introduction, we use the chain model shown in Fig. 1 instead of the current Internet standard (shell model). This allows us to exploit the chronological ordering of signatures as e.g. provided by FSS. In the chain model all signatures in the chain are validated at the time of their generation, meaning revocation and certificate validity is checked for that time. To describe the shell model to analyze other scenarios, different inference rules have to be defined. For a detailed discussion of validity models see e.g. [3,7].

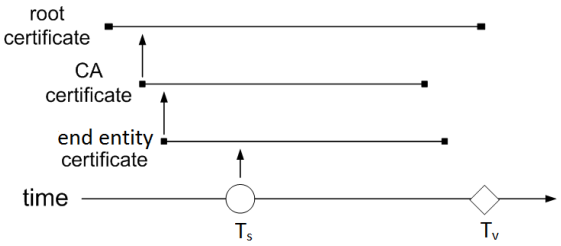

Fig. 1. Chain Model
Signature Generation at T_s, Signature Verification at T_v. Vertical arrows show the point in time used for validation of the superordinate certificate.

Definition 2 (Inference Rules). *Statements can be derived from an existing* View^t *according to the following rules:*

Certificate Validity $\forall\ \mathrm{CA}, \mathcal{A}, i_r \leq i_v, i_c \in \mathcal{I}_1, i_v \in \mathcal{I}_2$: $\mathsf{Trust}(\mathrm{CA}, \mathcal{I}_1)$, $\mathsf{Cert}(\mathrm{CA}, \mathcal{A}, i_c, \mathcal{I}_2), (\neg\mathsf{Revoc}(\mathrm{CA}, C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2}, i_r)) \vdash \mathsf{Valid}(C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2}, i_v)$
Signature Validity $\forall\ \mathrm{CA}, \mathcal{A}, D, i_s \in \mathcal{I}_2$:
$\mathsf{Valid}(C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2}, i_s), \mathsf{Signature}(\mathcal{A}, D, i_s) \vdash \mathsf{Valid}(S_{\mathcal{A},D,i_s})$

Note, to extend the model to certificate chains of arbitrary length, one simply considers certificates as signed documents while processing the chain, and derives the validity accordingly.

So far our model is static. To allow the definition of non-repudiation we introduce *transitions* between Views. The transitions model that new information enters a users View in form of certificates, signatures or revocation information. Besides that, a user might trust a new (Sub-)CA.

Definition 3 (Time & Transitions). *Let* View^t *be the* View *at time* t *and* $\mathsf{View}^t \xrightarrow{\texttt{trans}} \mathsf{View}^{t+1}$ *denote the transition from* View^t *to* View^{t+1}*. Let* CA *denote a (Sub-)CA,* $\mathcal{A}$ *an end-entity's identity,* D *a document and* $\mathcal{I}$ *an interval. We allow the following four transitions between* Views*:*

- $\mathsf{View}^t \xrightarrow{\mathsf{sign}(\mathcal{A},D,i)} \mathsf{View}^{t+1}$ *adds* $\mathsf{Signature}(\mathcal{A}, D, i)$ *to* View*.*
- $\mathsf{View}^t \xrightarrow{\mathsf{issue}(\mathrm{CA},\mathcal{A},i,\mathcal{I})} \mathsf{View}^{t+1}$ *adds* $\mathsf{Cert}(\mathrm{CA}, \mathcal{A}, i, \mathcal{I})$ *to* View*.*

– $\mathsf{View}^t \xrightarrow{\mathsf{trust}(\mathrm{CA},\mathcal{I})} \mathsf{View}^{t+1}$ *adds* $\mathsf{Trust}(\mathrm{CA},\mathcal{I})$ *to* View.
– $\mathsf{View}^t \xrightarrow{\mathsf{revoke}(\mathrm{CA},C_{\alpha,\beta,\gamma,\varepsilon},i)} \mathsf{View}^{t+1}$ *adds* $\mathsf{Revoc}(\mathrm{CA},C_{\alpha,\beta,\gamma,\varepsilon},i)$ *to* View.

Please note that derived statements are temporary. After a transition between two views, the inference rules are used again, to obtain the full set of statements. With $\overline{\mathsf{View}}$ we denote the set of all statements that can be inferred from View. So, if $\mathsf{stmt} \in \mathsf{View}^t$ it does not have to be the case that $\mathsf{stmt} \in \overline{\mathsf{View}^{t'}}$ for $t \neq t'$. For example, if a certificate gets revoked, Valid might be inferable beforehand but not after Revoc has been added to the View.

2.2 Non-Repudiation

In this work we assume adversaries that try to break the non-repudiation property and to which we refer as *repudiation adversaries*. The goal of such an adversary is to retroactively invalidate a signature, validly generated by herself. The adversary has access to the corresponding key pair, including the secret key and the certificate. The adversary might also have different other key pairs and certificates, possibly issued by other CAs. We give the classic non-repudiation definition [12] in our model. This allows a more precise analysis of repudiation adversaries.

Definition 4 (Non-Repudiation). *A PKI offers non-repudiation if the following implication is always true, even in presence of a malicious end-entity that might sign arbitrary messages, request new certificates and ask any CA to revoke any of her certificates at anytime.*

$$\forall\ i,\ t \leq t'\ :\ \mathsf{Valid}(S_{\mathcal{A},D,i}) \in \overline{\mathsf{View}^t} \Rightarrow \mathsf{Valid}(S_{\mathcal{A},D,i}) \in \overline{\mathsf{View}^{t'}}.$$

We briefly discuss the implications of this definition. The left part of the implication – $\mathsf{Valid}(S_{\mathcal{A},D,i}) \in \overline{\mathsf{View}^t}$ implies that

$$\{\mathsf{Signature}(\mathcal{A},D,i), \mathsf{Trust}(\mathrm{CA},\mathcal{I}_1), \mathsf{Cert}(\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2)\} \subseteq \overline{\mathsf{View}^t}$$

with $i_c \in \mathcal{I}_1,\ i \in \mathcal{I}_2$ according to the previously given inference rules and definitions. Furthermore, $\mathsf{Revoc}(\mathrm{CA},C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2},i_r) \notin \overline{\mathsf{View}^t}$ for all $\mathcal{I}_2 \ni i_r \leq i$. In other words, three things must be in View^t: (i) trust in the certification authority CA that issued the end-entity certificate for the document signing entity $\mathcal{A}$, (ii) the certificate of $\mathcal{A}$ that has been issued while CA has been trusted, (iii) a signature on the verified document D that has been issued by the end-entity $\mathcal{A}$ while his certificate has been valid, i.e. was not revoked or expired. The right part of the implication only differs in the time of inference of the Valid statement. Thus, everything above must hold for all future points in time t'.

Accordingly, the goal of the adversary is to produce a valid document signature $\mathsf{Signature}(\mathcal{A},D,i)$ such that there exists a point in time t' where the signature is verified as invalid, after it has been verified as valid. Therefore, we define backdated revocation and show, that its prevention implies non-repudiation and vice versa in the chain model.

Definition 5 (Backdated Revocation). *Let* View^t *be the* View *at time* t *and* View^{t+1} *denote the view after a transition. According to the revocation transition, backdated revocation is defined as:*
$\mathsf{View}^t \xrightarrow{\mathsf{revoke}(\mathrm{CA}, C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I},i_r})} \mathsf{View}^{t+1}$, *if* $\exists\ \mathsf{View}^{t^*} \ni \mathsf{Valid}(S_{\mathcal{A},D,i_s})$, *with* $t^* \leq t \wedge i_s \geq i_r$.

Theorem 1 (Non-Repudiation $\Leftrightarrow$ No Backdated Revocation). *A PKI offers non-repudiation according to Definition 4 if and only if it does not allow backdated revocation according to Definition 5.*

Proof. $\Leftarrow$: If there was a successful repudiation attack, then there must exist two Views $\overline{\mathsf{View}^t} \supseteq \{\mathsf{Valid}(S_{\mathcal{A},D,i_s}), \mathsf{Trust}(\mathrm{CA},\mathcal{I}_1), \mathsf{Cert}(\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2)\}$ and $\overline{\mathsf{View}^{t'}} \supseteq \{\mathsf{Trust}(\mathrm{CA},\mathcal{I}_1), \mathsf{Cert}(\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2), \mathsf{Revoc}(\mathrm{CA}, C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2}, i_r)\}$, with $t \leq t', i_r \leq i_s$. As $\mathsf{Valid}(S_{\mathcal{A},D,i_s})$ is contained in $\overline{\mathsf{View}^t}$, it can not contain $\mathsf{Revoc}(\mathrm{CA}, C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2}, i_r)$. Hence, $\mathsf{Revoc}(\mathrm{CA}, C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2}, i_r)$ must have been added later, which exactly corresponds to Definition 5.
$\Rightarrow$: If the PKI allows backdated revocation, the adversary is allowed to ask CA to add $\mathsf{Revoc}(\mathrm{CA}, C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2}, i_r)$ with $i_r \leq i_s$ to the $\mathsf{View}^{t'}$. □

3 Enabling Non-Repudation

Theorem 1 directly shows the impossibility to achieve non-repudiation if backdated revocation is allowed. Hence, a PKI that offers non-repudiation *must not* allow backdated revocation. However, when considering the facets of backdated revocation there are different security goals that contradict each other. This conflict needs to be resolved, and is discussed in the following.

Contradicting Security Goals. We have learned from our formal model, that there is no way to guarantee non-repudiation in case backdated revocation is allowed. On the other hand, backdated revocation is required in certain scenarios, namely whenever it is possible that the signature key might get compromised and maliciously used without being noticed immediately by the key owner. For example consider a classical setup for digital signatures where the private key is stored on the user's machine (e.g. PC). Here, the detection of a key compromise may take some time in which the adversary who stole the key may already have generated signatures. The phase between key compromise and the observation of the compromise (followed by the revocation) is called *gray phase*. The gray phase in this case however can be very long, such that it is clearly impossible to prohibit backdated revocation. This is because backdated revocation is required to invalidate the signatures generated by the adversary before the compromise detection.

Thus, in order to be able to prohibit and subsequently prevent backdated revocation, scenarios with a (possibly large) gray phase must be excluded or the gray phase must be eliminated by technical means. Therefore, in scenarios that require non-repudiation, the secret key has to be protected in a way that prevents unnoticed compromises. A common solution that allows for a minimal

gray phase is to store keys on smart cards, trusted platform modules (TPM) etc. Private keys are not extractable from these devices and can only be used when the according secret, e.g. PIN, is known. This allows for the assumption of "immediate" key compromise detection and subsequently the prohibition of backdated revocation in the sense that an adversary is not able to immediately crack the additional secret (e.g. PIN) and thus use the stolen key before the key compromise, i.e. disappearance of the key storage device, is detected. Note, that in Section 5 we show how our solution supports immediate compromise detection for end-entities. This further justifies the assumption of a marginal gray phase. In general, backdated revocation is not necessary when the gray phase is marginal.

How to prevent Backdated Revocation. With the aforementioned it is sufficient and legitimate to prohibit backdated revocation. Nevertheless, the question remains how to enforce the prohibition in practice, i.e. how to implement our model. We stick to the PKI setting and hence we assume, that the CA as TTP is the only entity that can revoke certificates. Therefor it must be possible for the CA to ensure that the revocation only invalidates signatures that were not assumed to be valid before. This can either be done by explicitly defining the views $View^t$ and binding each signature, as well as the revocation, to a certain view, using a globally visible and unique index t for views and signatures. Or second, to define $View^t$ implicitly by maintaining a local chronological order among the signatures made with one key using a local immutable index. By binding the revocation to the current index, it is correctly linked into the chronological order of the signatures. Note, that this requires the responsible CA to know the correct current local index.

The first approach is followed by the application of time-stamps, using real time as the global index. The time-stamps are generated by TTPs called Time-Stamping Authorities (TSA). This approach is widely accepted to be correct but also comes with a lot of inefficiencies. To evaluate our model, we proof that the time-stamping approach allows to implement our model s.th. the resulting PKI provides non-repudiation. The proof can be found in Appendix A together with a discussion of its inherent inefficiencies.

For our new solution we use the second approach of local indexing, which can be realized using forward secure signature schemes. This is described in the following section.

4 Sign & Report

In this section, we propose our solution applying local indexing to enforce non-repudiation using forward secure signature schemes (FSS). It does not have the disadvantages and inefficiencies of the TSA approach (cf. Appendix A). Furthermore, we show how compromise detection for end-entities can easily be incorporated to prevent gray periods.

As some reader might not be familiar with the concept of FSS, we first informally recall the related definitions and discuss some properties of such schemes

more detailed. For a formal treatment we refer the reader to [4]. We will not discuss the definitions regarding traditional signature schemes like existential unforgeability under chosen message attacks (EUF-CMA). Here we refer the reader to [11]. Forward security can only be achieved using key evolving signature schemes, which will be explained first. Afterwards, we show how to implement the PKI model using FSS, such that backdated revocation can efficiently be prevented.

Key Evolving Signature Schemes. Once generated, a key pair remains unchanged for the whole lifetime, in a traditional signature scheme (SIG). In contrast, the secret key $\mathtt{sk}$ changes over time, while the public key $\mathtt{pk}$ remains the same, in a key evolving scheme (KES). More specific, the lifetime of a key pair is split into several time-periods, say T. The number of time-periods T becomes a public parameter of a KES and is taken as an additional input by the key generation algorithm. The key generation algorithm outputs $(\mathtt{sk}_0, \mathtt{pk})$, where $\mathtt{sk}_0$ is the first secret key. In contrast to SIG, a KES has an additional key update algorithm, which updates the secret key at the end of each time period. The sign algorithm takes as an additional input an index of a time-period. This index also becomes part of the signature and is therefore also available for the verify algorithm. Finally, if a user generates a valid key pair and the key update algorithm is called at the end of each time-period, then a signature generated with the current secret key and the index of the current time period can be verified by any user with the corresponding public key.

Forward Secure Signature Schemes. A forward secure signature scheme (FSS) is a KES that provides the forward security property. The main idea behind forward security is that even after a key compromise all signatures created before should remain valid. The forward security property guarantees, that an adversary that is allowed to launch a chosen message attack for each time-period and learns the secret key of an adaptively chosen time-period i is unable to produce a valid forgery for time-period $j < i$. Note that forward security implies the standard notion of EUF-CMA extended to the case of KES. To make use of the forward security property in practice, a certificate is revoked from the index of the current time period onwards instead of revoking the complete certificate [7] in the case of key compromise.

Defining the time periods for FSS. One issue with FSS in practice is how the key update algorithm is triggered. It can either be called manually by the user, scheduled to run at the end of the time period, or be part of the signature algorithm, depending on the way the time periods are defined. Time periods can be defined in terms of time, e.g. one time period corresponds to one day, or the number of created signatures, i.e. a time period ends after the key was used to create a certain number of signatures. In the first case, the key update algorithm must be triggered periodically. This can only be automated on systems that have an internal clock and that are running each time an update is necessary. On smart cards, which are the common place to store end-entity signature keys, a manual update is required. This seems impractical. As an on-time key update is

required to preserve forward security, real time based time periods are problematic in practice. FSS schemes based on the number of generated signatures do not have these problems. Key update can be performed automated, based on a counter contained in the key holding device. Yet, in this case, the key indices are not linked to real time, which complicates correct revocation in practice as the index at the time of compromise must be known. However, our solution shows that this is achievable compared to the key update problem. Thus, we consider FSS where the periods are based on the number of signatures, namely an FSS, that evolves the key after each signature generation, as e.g. XMSS [8].

Sign & Report. We now show how to use a FSS, where a key update is performed after each signature, to implement a PKI according to our model, that efficiently prevents backdated revocation. If we used an FSS with real time based key update, the implementation would be straight forward, similar to the time-stamping approach. As we use an FSS where the key update occurs after each signature because of the discussed drawbacks of real time based key updates, things are a bit different. The validity periods of certificates, as well as the time interval for the trust relation are now described on the basis of indices. In general, these intervals will be $[0, T-1]$ for a key pair with T time periods. The indices used in Cert, Signature, Revoc and Valid statements correspond to key indices. For the first two types of statements, it is the current index of the key pair used to generate the signature. For the Revoc statement, it is the index starting from which on a key pair is revoked and for the last one it is the index at which revocation is checked. The indices of the views use real time. Please note that it is possible to additionally use real time validity periods, if this is required, e.g. for business purposes.

To prevent backdated revocation in such a PKI, a revocation must include the current index of an end-entity's key pair. Therefore the responsible CA must know this current index and be able to verify the correctness of this index. To achieve this, we define a Sign & Report approach for FSS.

Definition 6 (Sign & Report PKI). *A Sign & Report PKI implements the model defined in Section 2 replacing the abstract indices and intervals as described above. Let* R *denote a trusted third party in the PKI, e.g. a CA, which is responsible (and exclusively able) to issue the revocation of an end-entity $\mathcal{A}$'s certificate $C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}}$, when requested by $\mathcal{A}$. Whenever $\mathcal{A}$ generates a signature, the used key index i^* is reported to* R *that stores i^*. On input of revocation request by $\mathcal{A}$,* R *publishes* $\mathsf{Revoc}(\mathrm{CA}, C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}}, i^*+1)$.

We next show that a Sign & Report PKI provides non-repudiation, assumed that the index reporting is secure.

Theorem 2 (Sign & Report PKIs provide non-repudiation). *A Sign & Report PKI as defined above provides non-repudiation according to Definition 4.*

Proof. If the index reporting is implemented in a secure way, i.e. it is not possible for an end-entity to manipulate the reporting, backdated revocation is efficiently prevented. This is the case, because the index used for revocation is greater than

any index used by this end-entity before. The non-repudiation property follows from Theorem 1. □

Key Compromise Detection. The Sign & Report PKI makes it easily possible to monitor key usage and support end-entities in the detection of malicious key usage and subsequent revocation. Therewith, the justification for immediate revocation can be strengthened. The detection is made possible as, different from the TSA approach, for each end-entity a specific TTP is responsible.

Definition 7 (Sign & Report PKI with Compromise Detection). *A Sign & Report PKI with Compromise Detection is a Sign & Report PKI as defined in Definition 6 with the additional measure, that* R *requests a reconfirmation from* $\mathcal{A}$ *using an independent channel, whenever a new key index* i^* *is reported.* R *only accepts the new index if the reconfirmation succeeds. Otherwise,* R *publishes* $\mathsf{Revoc}(\mathrm{CA}, C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}}, i^*)$

5 Implementing Sign & Report

In this section we present a practical implementation of the Sign & Report approach from above. More specifically we show how to securely implement the index reporting, as everything else is straight forward. Note that we apply an FSS, namely XMSS, that evolves the key after each signature generation. Thus, each signature is directly linked to a unique index. We also present extensions that allow protection from undetected key access and a pay per use pricing model for certificates. As discussed before, we assume that the private key is stored on a smart card and therefore can assume immediate revocation (cf. Section 3). With this preliminary, we define an online protocol to prevent backdated revocation.

Index Reporting Protocol. The basic idea of Sign & Report is that the current index is reported to the CA (or some TTP that maintains the revocation) immediately after signature generation. This requires an online step, yet seams reasonable, as most of today's computers are nearly always online. Thereby, the CA is enabled to keep track of the signing index and prevent the key owner from backdated revocation and repudiation of signatures. Thereby, it does not matter who reports the index, the verifier or the signer itself and this might be chosen depending on the specific application. In the first case, reporting can be performed in one combined step during online revocation status checking and would reflect the natural ambition of the verifier to obtain non-repudiation. The second case is desirable when the verifier is offline. However, then the signer needs to be able to prove the reporting. This can be realized by a *validity token* obtained from the CA and additionally serving as a proof for the absence of a revocation. Thus, additional revocation checking is made obsolete.

Figure 2 shows the protocol for the second case, but the adaptation to the first is straight forward. After signature generation (step (a)) the signature is sent to the CA (step (b)). The CA checks the signature for validity (step(c)) and generates a *validity token* for the signature index to confirm the logging. Herein, the signature verification ensures, that the report and confirmation of

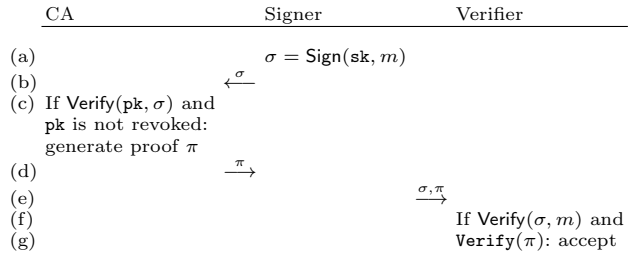

Fig. 2. Index Reporting Protocol

wrong index information is prevented. The token is sent back to the signer and subsequently transmitted (together with the signature) to the verifier (steps (d)-(e)), who can now validate the signature and the token.

By the index, the validity token is bound to a specific signature. Thus, it can be used for all future verifications without further online requests. Additionally, due to the forward security, the token for a certain index i can serve as a validity token for all preceding indices. Thus, if several signatures have to be validated, the logging request can be aggregated to only one, by requesting the token for the highest index.

The validity token can be realized as public key - index pair signed by the CA. One drawback of such a signed validity token is that the signature on the token has to be validated in addition to the validation of the end-entity signature. Furthermore, the size of the token is not optimal and signature generation is computationally complex. We propose to apply the Merkle tree variant of the Novomodo system [16] to generate the validity token (cf. Appendix B for details).

Incorporation of Compromise Detection. In the index reporting protocol, the issue of compromise detection and the prevention of gray periods can be addressed by adding an additional reconfirmation procedure for signature generation. Before confirming the logging of the index, the CA can request a reconfirmation from the key owner. A possibility to do so is to apply mobile transaction numbers as known from eBanking or similar to the usage presented in [6]. This helps to detect malicious key usage, as the key owner is informed about key usage via an independent channel. By additionally sending the document itself to the CA, the signed document could be sent back and displayed on a smartphone for verification. Therewith, undetected usage of the key is significantly less probable, and even cases where e.g. the smart card is left unwatched for a certain time period and malicious key usage does not necessarily involve the observed loss of the card can be detected.

Efficiency. Besides cutting down on the maintenance of an additional and independent TSA infrastructure and the overhead of time-stamp validation, our approach saves one online request considering the whole process from signature generation to verification. While in the TSA approach the time-stamp and the revocation status need to be requested, one online request, namely the index logging request, is sufficient in our approach.

6 Conclusion

In this work we showed how to extend the existing formal models for a PKI such that it becomes possible to describe the non-repudiation property. Using our model, we proved that non-repudiation is achieved, if and only if backdated revocation is prevented (assuming CAs are trustworthy and the used cryptographic algorithms are secure). The main improvement of our model is that it is dynamic. It might also be useful in analyzing other properties of PKIs. Furthermore, we showed how to realize our model using FSS. We presented an index reporting protocol to implement this new approach. It has some clear advantages compared to today's solutions regarding computation costs and storage. Furthermore, it allows for a convenient integration of an additional reconfirmation step to detect compromises and improve the overall security. Besides that, the index reporting approach allows for another interesting application we shortly want to mention, namely the realization of a pay per use pricing model. This is possible, as the CA is enabled to monitor the frequency of key usage. Thus, the costs for a certificate can be spread over the whole key lifetime, therewith the costs to get a certificate can be lowered significantly. This might on the one hand help to decrease the initial barrier of buying a certificate for end-entities that rarely apply electronic signatures. On the other hand revenues for the CAs are generated at the time when the efforts arise for the management of certificates.

While Sign & Report improves the performance of document signatures it still requires one online step per signature, i.e. the signer has to communicate with a TTP during signature creation (This step can be shifted to the verifier, but it remains one online step per signature). We were unable to get rid of this costly step. Hence the question arises if it is possible at all to prohibit backdated revocation by using an "offline" solution, namely a solution where the end-entity does not report information about each signature to a TTP? As we assume the signer as well as the verifier to be potentially malicious, neither of them can be trusted. A trusted device in possession of the signer or verifier can also not be trusted, as it might in the long term be possible to tamper with or it might get destroyed. In both cases, i.e. using a local or a global index, the indexing – especially the order – and the linking between indices and signatures must be immutable. Furthermore, using local indexing the CA must be able to obtain the correct current value of the index used by the user in case of a revocation. Given these constraints it seems impossible to us to solve this problem without an online step. At least with existing techniques we see no solution to this challenge.

References

1. C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. RFC 3161 — Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161 (Proposed Standard), 2001. Updated by RFC 5816.
2. ANSI X9.95-2012 — Trusted Time Stamp Management and Security, 2012.
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. Springer, 2010.

4. M. Bellare and S. K. Miner. A Forward-Secure Digital Signature Scheme. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO'99, pages 431–448. Springer, 1999.
5. K. Bicakci, B. Crispo, and A. S. Tanenbaum. How to incorporate revocation status information into the trust metrics for public-key certification. In *Proceedings of the 2005 ACM symposium on Applied computing*, SAC'05, pages 1594–1598. ACM, 2005.
6. J. Braun, M. Horsch, and A. Wiesmaier. iPIN and mTAN for secure eID applications. In *Proceedings of the 8th international conference on Information Security Practice and Experience*, ISPEC'12, pages 259–276. Springer, 2012.
7. J. Braun, A. Hülsing, A. Wiesmaier, M. A. G. Vigil, and J. Buchmann. How to avoid the breakdown of public key infrastructures - forward secure signatures for certificate authorities. In *9th European PKI Workshop: Research and Applications*, EuroPKI'12, 2012.
8. J. Buchmann, E. Dahmen, and A. Hülsing. XMSS - A practical forward secure signature scheme based on minimal security assumptions. In *Proceedings of the 4th international conference on Post-Quantum Cryptography*, PQCrypto'11, pages 117–129. Springer, 2011.
9. 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.
10. F. Elwailly, C. Gentry, and Z. Ramzan. QuasiModo: Efficient Certificate Validation and Revocation. In *Public Key Cryptography*, PKC'04, pages 375–388. Springer, 2004.
11. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
12. ISO/IEC 13888-1 — Information technology-security techniques-non-repudiation, Part 1, 1997.
13. ISO/IEC 18014 — Information technology – Security techniques – Time-stamping services, 2009.
14. J. Marchesini and S. Smith. Modeling public key infrastructures in the real world. In *Proceedings of the Second European conference on Public Key Infrastructure*, EuroPKI'05, pages 118–134. Springer, 2005.
15. U. M. Maurer. Modelling a Public-Key Infrastructure. In *Proceedings of the 4th European Symposium on Research in Computer Security: Computer Security*, ESORICS'96, pages 325–350. Springer, 1996.
16. S. Micali. NOVOMODO: Scalable Certificate Validation and Simplified PKI Management. In *1st Annual PKI Research Workshop*, PKI'02, pages 15–25, 2002.
17. The European Parliament and the Council of the European Union. Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures. In *European Union law*. The European Parliament and Council, 1999.
18. J. Zhou, F. Bao, and R. Deng. Minimizing ttp's involvement in signature validation. *International Journal of Information Security*, 5(1):37–47, 2006.

A Application of Time-Stamping Authorities

To evaluate our model, in this section we show that the common approach of time-stamping allows to implement a PKI that provides non-repudiation. To implement a PKI according to the above model using time-stamping, a trusted third party called Time-Stamping Authority (TSA) adds a trusted time to each signature. This can be implemented in different ways [1,2,13]. The most common one is to sign the signature together with a time-stamp. The signature time of a document signature as well as the issuance time of a certificate are given by a time-stamp. Validity periods of certificates are defined by the issuing CA and the time interval for the trust statement is defined by the user (or according to the certificates of higher levels in the hierarchy). Both are defined in terms of real time. If a CA is asked to revoke a certificate, it uses the current time as revocation index. All views are also bound to real time. So all indices and intervals are directly linked to real time and an order is defined according to the calendar.

Definition 8 (Sign & Stamp PKI). *A Sign & Stamp PKI implements the above model using the real time for all indices and intervals. In a Sign & Stamp PKI, every signature is time-stamped by a trusted third party called time-stamping authority (TSA). If a CA is asked to revoke a certificate, it uses the current time as revocation index.*

In a Sign & Stamp PKI times in statements and Views have a strictly monotonic increasing order, e.g. a signature generated at time t can not be in a $\mathsf{View}^{t'}$ with $t' < t$. As revocations always include the current time, backdated revocation is impossible (recall that we assume the used signature schemes to be perfectly secure). The following theorem follows immediately from Theorem 1 and the fact that a Sign & Stamp PKI prevents backdated revocation.

Theorem 3 (Sign & Stamp PKIs provide Non-Repudation). *A Sign & Stamp PKI according to Definition 8 provides non-repudiation according to Definition 4.*

Disadvantages of the TSA approach. We have seen that the TSA approach enables non-repudiation. Yet, it comes with many disadvantages. First, the setup and maintenance of an additional and independent TSA infrastructure and the trustworthiness of the TSAs to apply the correct date and time is required. Second, it introduces overhead during signature generation (for the online request and generation of the proof of existence) and during validation (for the proof validation). Third, storage overhead is introduced, i.e. time-stamps or MACs must be stored, in addition to the signature itself, or a huge amount of transient keys must be managed. Database based approaches require the central storage of all issued signatures. And fourth, time-stamps relying on electronic signatures themselves 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. On the other hand, database based approaches solely rely on the security of the central database.

B Validity Tokens using Novomodo

Instead of using hash chains as in the basic Novomodo system [16] one can use Merkle hash trees. This significantly improves the verification time. To realize the validity tokens, the root of a Merkle tree with T leaves (where T is the number of periods, the respective key pair is valid for) is included into the certificate. If the certificate is valid in period i, the CA releases the leaf at position i and the siblings on the path to the root. QuasiModo trees [10] allow even smaller trees and on average shorter paths to the tree root by using interior nodes, yet for standard Merkle trees highly efficient traversal methods are available [8]. The validity tokens of the revocation tree can be generated in a pseudorandom fashion, comparable to the approach used for XMSS key generation [8]. An application of the hash function gives us the leaves from which the Merkle tree is computed. Furthermore, the tree traversal algorithm from [8] can be used to evenly split the computational effort over all periods. To prevent delays, a certain number of validity tokens can be precomputed and stored, reducing the effort to a table lookup during the online request.