Digital Signatures for e-Government – A Long-Term Security Architecture*

Przemysław Błaśkiewicz, Przemysław Kubiak, and Mirosław Kutyłowski

Institute of Mathematics and Computer Science,
Wrocław University of Technology
{przemyslaw.blaskiewicz,przemyslaw.kubiak,miroslaw.kutylowski}@pwr.wroc.pl

Abstract. The framework of digital signature based on qualified certificates and X.509 architecture is known to have many security risks. Moreover, the fraud prevention mechanism is fragile and does not provide strong guarantees that might be regarded necessary for flow of legal documents.

Recently, mediated signatures have been proposed as a mechanism to effectively disable signature cards. In this paper we propose further mechanisms that can be applied on top of mediated RSA, so that we obtain signatures compatible with the standard format, but providing security guarantees even in the case when RSA becomes broken or the keys are compromised. Our solution is well suited for deploying a large-scale, long-term digital signature system for signing legal documents. Moreover, the solution is immune to kleptographic attacks as only deterministic algorithms are used on user's side.

Keywords: mRSA, PSS padding, signatures based on hash functions, kleptography, deterministic signatures, pairing based signatures.

1 Introduction

Digital signature seems to be the key technology for securing electronic documents against unauthorized modifications and forgery. However, digital signatures require a broader framework, where cryptographic security of a signature scheme is only one of the components contributing to the security of the system.

Equally important are answers to the following questions:

- how to make sure that a given public key corresponds to an alleged signer?
- how to make sure that the private signing keys cannot be used by anybody else but its owner?

While there is a lot of research on the first question (with many proposals such as alternative PKI systems, identity based signatures, certificateless signatures), the second question is relatively neglected, despite that we have no really good answers for the following specific questions:

* The paper is partially supported by Polish Ministry of Science and Higher Education, grant N N206 2701 33, and by "MISTRZ" programme of Foundation for Polish Science.

X. Lai et al. (Eds.): E-Forensics 2010, LNICST 56, pp. 256–270, 2011.

1. how to make sure that a key generated outside a secure signature-creation device is not retained and occasionally used by the service provider?
2. how to make sure that an unauthorized person has not used a secure signature-creation device after guessing the PIN?
3. if a secure signature-creation device has no keypad, how to know that the signatures under arbitrary documents are created by the PC in cooperation with the signature creation device?
4. how to make sure that there are no trapdoors or just security gaps in secure signature-creation devices used?
5. how to make sure that a secure signature-creation device is immune to any kind of physical and side-channel attacks? In particular, how to make sure that a card does not generate faulty signatures giving room for fault cryptanalysis?
6. how to check the origin of a given signature-creation device, so that malicious replacement is impossible?

Many of these problems are particularly hard, if signature creation devices are cryptographic smart cards. Some surrogate solutions have been proposed:

ad 1) Retention of any such data has been declared as a criminal act. However, it is hard to trace any activity of this kind, if it is carefully hidden. Technical solutions, such as distributed key generation procedures have been proposed, so that a card must participate in key generation and the service provider does not learn the whole private key. However, in large scale applications these methods are not very attractive due to logistics problems (generation of keys at the moment of handing the card to its owner takes time and requires few manual operations).

ad 2) Three failures to provide a PIN usually lead to blocking the card. However, the attacker may return the card after two trials into the wallet of the owner and wait for another chance. This is particularly dangerous for office applications.

ad 3) This problem might be solved with new technologies for inputing data directly to a smart card. Alternatively, one may try to improve security of operating systems and processor architecture, but it seems to be extremely difficult, if possible at all.

ad 4) So far, a common practice is to depend on declarations of the producers (!) or examinations by specially designated bodies. In the latter case, the signer is fully dependant on honesty of the examiner and completeness of the verification procedure. So far, the possibilities of thorough security analysis of chips and trapdoor detection are more a myth than technical reality. What the examiner can do is to check if there are some security threats that follow from violating a closed set of rules.

ad 5) Securing a smart card against physical attacks is a never ending game between attacking possibilities and protection mechanisms. Evaluating the state of the art of attacking possibilities as well as effectiveness of hardware protection requires insider knowledge, where at least part of it is an industrial secret. So it is hard to say whether declarations of the manufacturers are dependable or, may be, they are based on their business goals.

ad 6) The main protection mechanism remains the protection of a supply chain and visual protection mechanisms on the surface of the card (such as holograms). This is effective, but not against powerful adversaries.

Kleptographic Channels. In the context of securing signature creation devices we especially focus on kleptographic attacks [1,2]. Kleptography is a set of cryptographic techniques that allow implementation of a kleptographic side channel within the framework of a randomized cryptographic protocol. Such channel is visible and usable only for its creator. Information transmitted in the channel is protected by a "public" key (i.e. asymmetric key used solely for encryption), information retrieval is possible with a matching "private" key. Let us assume that a manufacturer has planted a kleptographic channel in a batch of devices he produced. Then physical inspection of the tampered devices and extracting the "public" key do not give access to information hidden in the kleptographic channel of this or any other device.

There are techniques of setting a kleptographic channel in nondeterministic crypto protocols in such a way that the protocol runs according to the specification, the statistical properties of its output *are not altered*, and, on top of that, the time characteristics remain within acceptable interval [3]. In the case of a nondeterministic signature, the information can be hidden in the signature itself. For deterministic protocols (like RSA for example) the nondeterministic part is the key generation, so the information may be hidden there (for details see e.g. [4], [5]).

Mediated Signatures as Secure Signing Environment. The idea of mediated signatures is that signature creation requires not only using a private signing key, but also an additional key (or keys) held by a security mediator or mediators (SEM). Particularly straightforward is constructing mediated signatures on top of RSA. The idea is to split the original private key and give its parts to the signer and the mediator. It can be done in an additive way ([6,7,8]), or a multiplicative way ([7,9]). We focus on the former variant, because it broadens the set of ready-to-use algorithms for distributed generation of RSA keys and facilitates the procedure described in Sect. 4. Specifically, if d is the original private key, then the mediator gets $d - d_u$ and the signer gets d_u, where d_u is (pseudo)random generated and distributed according to private keys regime.

The idea presented in [8] is to use mediated signatures as a fundamental security mechanism for digital signatures. The mediator is located at a central server which keeps a black list of stolen/lost signature cards and refuses to finalize requests from such cards. Therefore, a withheld card cannot create a signature, even if there is no mechanism to block the card itself. It also allows for temporary disabling a card, for instance outside the office hours of the signer or just on request of the owner. Note that the mediator can also monitor activity of the card for accordance with its security policy (e.g. a limited number of signatures per day). Moreover, in this scenario recording of the time of the signature can be provided by the mediator which is not possible in the traditional mode of using signature cards.

1.1 Our Contribution

We propose a couple of additional security mechanisms that are backwards compatible: standard software can verify such signatures in the old way. We address the following issues:

- protection against kleptographic attacks on RSA signatures exploiting padding bits [5],

– combining RSA signature with a signature based on discrete logarithm problem, so that in case of breaking RSA a forged signature can be recognized,
– a method of generating signatures between the signer and the mediator, so that a powerful adversary cannot create signatures even if he knows the keys.

This paper is not on new signature schemes but rather on system architecture that should prevent or detect any misuse of cryptographic mechanisms.

2 Building Blocks

2.1 RSA Signatures and Message Encoding Functions

An RSA signature is a result of three functions: a hash function h applied to the message m to be signed, a coding function C converting the hash value to a number modulo RSA number N, and finally an exponentiation modulo N:

$$(C(h(m)))^d \bmod N.$$

The coding function must be chosen with care (see attacks [10], [11]).

In this paper we use EMSA-PSS coding [12]. A part of the coding, important in tightening security reduction (cf. [13]), is encoding a random *salt* string together with the hash value. Normally, this may lead to many problems due to kleptographic attacks, but we shall use the *salt* as place for embedding another signature. Embedding a signature does not violate the coding – according to Sect. 8.1 of [12]: as *salt* even "a fixed value or a sequence number could be employed (...), with the resulting provable security similar to that of FDH" (Full Domain Hashing).

Another issue, crucial for the embedded signature, is the length of *salt*. In Appendix A.2.3 of [12] a type `RSASSA-PSS-params` is described to include, among others, a field `saltLenght` (i.e. octet length of the *salt*). [12] specifies the default value of the field to be the octet length of the output of the function indicated in the `hashAlgorithm` field. However, `saltLength` may be different: let $modBits$ denote bitlength of N, and $hLen$ denotes the length in octets of the hash function output, then the following condition (see Sect. 9.1.1 of [12]) imposes an upper bound for *salt* length:

$$\lceil (modBits - 1)/8 \rceil - 2 \geq \texttt{saltLength} + hLen.$$

2.2 Deterministic Signatures Based on Discrete Logarithm

Most discrete logarithm based signatures are probabilistic ones. The problem with these solutions is that there are many kleptographic schemes taking advantage of the pseudo-random parameters for signature generation, that may be potentially used to leak keys from a signature creation device. On the other hand, DL based signatures are based on different algebraic structures than RSA and might help in the case when security of RSA becomes endangered.

Fortunetely, there are deterministic signatures based on DL Problem, see for instance the BLS [14] or [15].

In this paper we use BLS: Suppose that $\mathbb{G}_1$, $\mathbb{G}_2$ are cyclic additive groups of prime order q, and let P be a generator of $\mathbb{G}_1$. Assume that there is an efficiently computable isomorphism $\psi : \mathbb{G}_1 \to \mathbb{G}_2$, thus $\psi(P)$ is a generator of $\mathbb{G}_2$. Let $\mathbb{G}_T$ be a multiplicative group of prime order q, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a non-degenerate bilinear map, that is:

1. for all $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}$, $\hat{e}([a]P, [b]Q) = \hat{e}(P, Q)^{ab}$, where $[k]P$ denotes scalar k multiplication of element P,
2. $\hat{e}(P, \psi(P)) \neq 1$.

For simplicity one may assume $\mathbb{G}_2 = \mathbb{G}_1$, and $\psi \equiv \mathrm{id}$. In the BLS scheme $\mathbb{G}_1$ is a subgroup of points of an elliptic curve E defined over some finite field $\mathbb{F}_{p^r}$, and $\mathbb{G}_T$ is a subgroup of the multiplicative group $\mathbb{F}^*_{p^{r\kappa}}$, where κ is a relatively small integer, say $\kappa \in \{12, \ldots, 40\}$. The number κ is usually called *the embedding degree*. Note that $q|\#E$, but for security reasons we require that $q^2 \not| \#E$.

The signature algorithm comprises of calculation of the first point $H(m) \in \langle P \rangle$ corresponding to a message m, and computing $[x_u]H(m)$, i.e. multiplication of elliptic curve point $H(m)$ by scalar x_u being the private key of the user making the signature. The signature is the x-coordinate of the point $[x_u]H(m)$. Verification of the signature (see Sect. 3) takes place in the group $\mathbb{F}^*_{p^{r\kappa}}$, and it is more costly than signature generation.

2.3 Signatures Based on Hash Functions

Apart from RSA and discrete logarithm based signatures there is a third family: signatures based on hash functions. Their main advantage is fast verification, their main disadvantage is limitation on the number of signatures one can create – basic schemes of this kind are usually one-time signatures. This drawback can be alleviated by employing *Merkle trees*, and the resulting schemes (*Merkle Signature Scheme – MSS*) offer multipe-time signatures. In this case however, the maximal number of signatures is determined at the time of key generation. This in turn causes complexity issues, since building a large, single Merkle tree is calculation demanding. In [16], the GMSS algorithm loosens this limitation: even 2^{80} signatures might be verified with the root of the main tree.

2.4 Overview of System Architecture

The system is based on security mediator SEM, as in [17]. However, we propose to split SEM into t sub-centers sub-SEM_i, $i = 1, \ldots, t$, $t \geq 2$ (such decomposition would alleviate the problems of information leakage from a SEM). System components on the signer's side are: a PC and a smart card used as a *secure signature creation device*.

When the signer wishes to compose a signature, then the smart card performes some operations in interaction with the SEMs. The final output of the SEMs is a high quality signature – its safety is based on many security mechanisms that on the whole address the problems and scenarios mentioned in the introduction.

3 Nested Signatures

Since long-term predictions about scheme's security are given with large amount of uncertainty, it seems reasonable to strengthen the RSA with another deterministic signature scheme — the BLS [14]. We combine them together using RSASSA-PSS, with the RSA signature layer being the mediated one, while BLS is composed solely by the smart card of the signer. Thanks to the way the message is coded the resulting signature can be input to a standard RSA verification software which will still verify the RSA layer in the regular way. However, software aware of the nesting can perform a thorough verification and check both signatures.

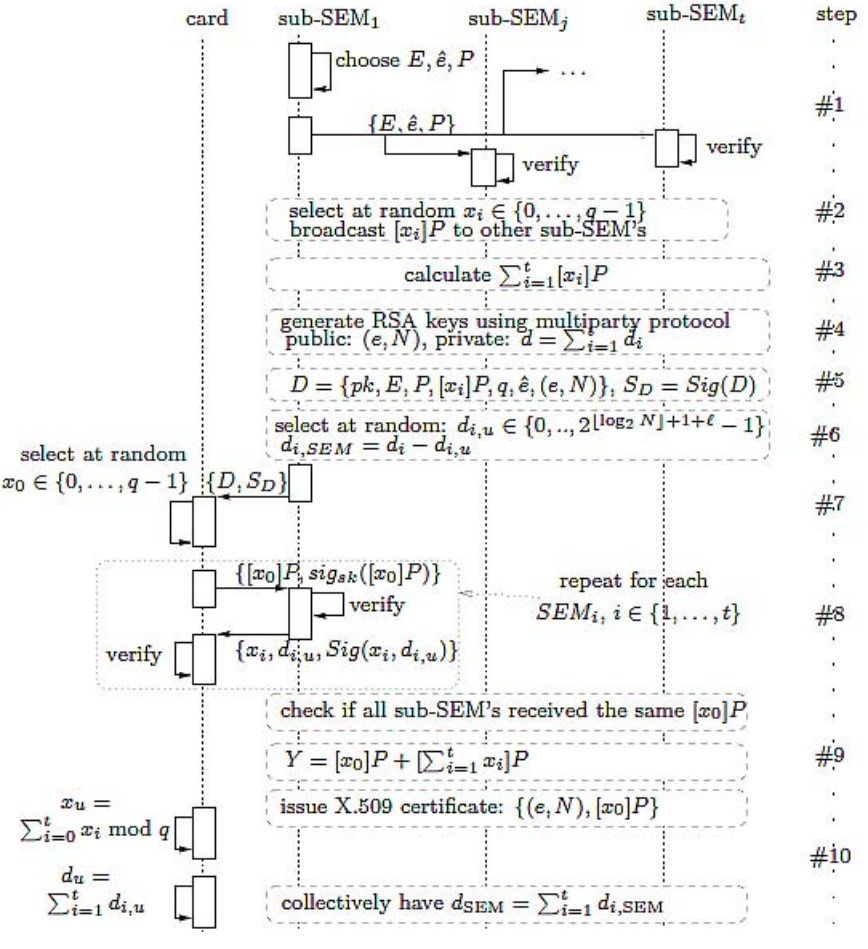

Fig. 1. Data flow for key generation. Operations in rounded rectangles are performed distributively.

Key Generation. We propose that the modulus N and the secret exponent d of RSA should be generated outside the card in a multiparty protocol (accordingly, we divide the security mediator SEM into t sub-SEMs, $t \geq 2$). This prevents any trapdoor or kleptography possibilities on the side of the smart card, and makes it possible to use high quality randomness. Last not least, it may speed up logistics issues (generation of RSA keys is relatively slow and the time delay may be annoying for an average user).

Multiparty generation of RSA keys has been described in the literature: [18] – for at least 3 participants (for real implementation issues see [19], for a robust version see [20]), [21] – for two participants, or a different approach in [22].

Let us describe the steps of generating the RSA and BLS keys in some more detail (see also Fig. 1):

Suppose that the card holds some single, initial, unique priate key sk (set by the card's producer) for deterministic one-time signature scheme. Let the public part pk of the key be given to SEM before the following protocol is executed. Assume also that the card's manufacturer has placed into the card SEM's public key for verification of SEM's signatures.

1. sub-SEM_1 selects an elliptic curve defined over some finite field (the choice determines also a bilinear mapping $\hat{e}$) and a basepoint P of prime order q. Then sub-SEM_1 transmits this data together with definition of $\hat{e}$ to the other sub-SEM's for verification.
2. If the verification succeeded, each sub-SEM_i picks $x_i \in \{0, \ldots, q-1\}$ at random and broadcasts the point $[x_i]P$ to other sub-SEMs.
3. Each sub-SEM calculates $\sum_{i=1}^{t}[x_i]P$, i.e. calculates $[\sum_{i=1}^{t} x_i]P$.
4. The sub-SEMs generate the RSA-keys using a multiparty protocol: let the resulting public part be (e, N) and the secret exponent be $d = \sum_{i=1}^{t} \tilde{d}_i$, where $\tilde{d}_i \in \mathbb{Z}$ is known only to sub-SEM_i.
5. All sub-SEMs now distributively sign all public data D generated so far, i.e.: the public one time key pk (which serves as identifier of the addressee of data D), the definition of the field, curve E, points P, $[x_i]P$, $i = 1, \ldots, t$, order q of P, map $\hat{e}$ and RSA public key (e, N). The signature might also be a nested signature, even with the inner signature being a probabilistic one, e.g. ECDSA (to mitigate the threat of klepto channel each sub-SEM might xor outputs from a few random number generators).
6. Let ℓ is a fixed element from the set $\{128, \ldots, 160\}$ (see e.g. the range of additive sharing over $\mathbb{Z}$ in Sect. 3.2 of [22], and Δ in S-RSA-DEL delegation protocol in Fig. 2 of [23]). Each sub-SEM_i, $i = 1, \ldots, t$ picks $d_{i,u} \in \{0, \ldots, 2^{\lfloor \log_2 N \rfloor + 1 + \ell} - 1\}$ at random and calculates integer $d_{i,\mathrm{SEM}} = \tilde{d}_i - d_{i,u}$. Note that $d_{i,u}$ can be calculated independently of N (e.g. before N), only the length of N must be known.
7. The card contacts sub-SEM_1 over a secure channel and receives the signed data D. If verification of the signature succeeds the card picks its random element $x_0 \in \{0, \ldots, q-1\}$, and calculates $[x_0]P$.
8. For each $i \in \{1, \ldots, t\}$ the card contacts sub-SEM_i over a secure channel and sends it $[x_0]P$ and $sig_{sk}([x_0]P)$. The sub-SEM_i verifies the signature and only then does it respond with x_i and $d_{i,u}$ and a signature thereof (a certificate for the sub-SEM_i signature key is distributiely signed by all sub-SEMs, and is transferred

to the card together with the signature). The card immediately checks x_i against P, $[x_i]P$ from D.

9. At this point all sub-SEMs compare the received element $[x_0]P \in E$ (i.e. they check if the sk was really used only once). If this is so, then the value is taken as ID-card's part of the BLS public key. Then the sub-SEMs complete calculation of the key: $E, P \in E, Y = [x_0]P + [\sum_{i=1}^{t} x_i]P$, and issue an X.509 certificate for the card that it possesses the RSA key (e, N). In some extension field the certificate must also contain card's BLS public key for the inner signature. The certificate is signed distributively. Sub-SEM_t now transfer the certificate to the ID-card.
10. The card calculates its BLS private key as $x_u = \sum_{i=0}^{t} x_i \bmod q$ and its part of RSA private key as integer $d_u = \sum_{i=1}^{t} d_{i,u}$. Note that the remaining part $d_{\mathrm{SEM}} = \sum_{i=1}^{t} d_{i,\mathrm{SEM}}$ of the secret key d is distributed among sub-SEMs, who will participate in every signing procedure initiated by the user. Neither he nor the sub-SEMs can generate valid signatures on their own.
11. The card compares the certificate received from the last sub-SEM with D received from the first sub-SEM. As the last check the card initializes the signature generation protocol (see below) to sign the certificate. If the finalized signature is valid the card assumes that d_u is valid as well, and removes all partial $d_{i,u}$ and partial x_i together with their signatures. Otherwise the card discloses all data received, together with their signatures.

Each user should receive a different set of keys, i.e. different modulus N for RSA system and a unique (non-isomorphic with the ones so far generated) elliptic curve for the BLS signature. This can minimize damages that could result by breaking both systems using adequately large resources.

Signature Generation

1. The user's PC computes the hash value $h(m)$ of the message m to be signed, and sends it to the smartcard.
2. the smartcard signs $h(m)$ using BLS scheme: the first point $H(h(m))$ of the group $\langle P \rangle$, corresponding to $h(m)$, is calculated deterministically, according to the procedure from [14] (alternatively, the algorithm from [24] might be used, complemented by multiplication by scalar $\#E/q$ to get a point in the subgroup of order q), next $H(h(m))$ is multiplied by the scalar x_u, which yields point $[x_u]H(h(m))$. The BLS signature of $h(m)$ is the x-coordinate $x([x_u]H(h(m)))$ of the point $[x_u]H(h(m))$. The resulting signature is unpredictable to both the card's owner as well as other third parties. We call this signature the *salt*.
3. Both $h(m)$ and *salt* can now be used by the card as variables in execution of RSASSA-PSS scheme: they just need to be composed according to EMSA-PSS [12] and the result μ can now be simply RSA-exponentiated.
4. In the process of signature generation, the user's card calculates the d_u'th power of the result μ of EMSA-PSS padding and sends it, along with the message digest $h(m)$ and the padding result itself, to the SEM. That is, it sends the triple $(h(m), s_u, \mu)$, where $s_u = \mu^{d_u} \bmod N$.
5. The sub-SEMs finalize the RSA exponentiation: $s = s_u \cdot \prod_{i=1}^{t} \mu^{d_{i,\mathrm{SEM}}} \bmod N$, thus finishing the procedure of RSA signature generation.

6. At this point a full verification is possible: SEM verifies the RSA signature, checks the EMSA-PSS coding – this includes *salt* recovering and verification of the inner signature (it also results in checking if the card had chosen *the first* possible point on the curve while encoding $h(m)$). If the checks succeed, the finalized signature is sent back to the user. A failure means that the card has malfunctioned or behaved maliciously – as we see, the system-internal verification is of vital importance.

Note that during the signature generation procedure the smartcard and sub-SEMs cannot use CRT, as in this case the factorization of N would be known to all parties. This increases signing time, especially on the side of the card. But, theoretically, this can be seen as an advantage. For example, the signing time longer than 10 sec. means that one cannot generate more than 2^{25} signatures over the period of 10 years; we therefore obtain an upper limit on power of the adversary in results of [25] and [13]. In fact the SEM might arbitrarily set a lower bound for the period of time it must pass between two consecutive finalizations of signatures of the same user. Moreover, if CRT is not in use, then some category of fault attacks is eliminated ([26,27]).

Signature Verification. For given m and its alleged signature s:

1. The verifier calculates $h(m)$ and the point $H(h(m)) \in \langle P \rangle$.
2. Given the RSA public key (e, N) the verifier first calculates $\mu = s^e \bmod N$, and checks the EMSA-PSS coding against $h(m)$ (this includes *salt* recovery).
3. If the coding is valid then, given BLS public key E, P, Y, q, and $\hat{e}$, the verifier checks the inner signature. From $salt = x([x_u]H(h(m)))$ one of the two points $\pm[x_u]H(h(m))$ is calculated, denote this point by Q. Next, it is checked whether the order of Q equals q. If it does, then the verifier checks if one of the conditions holds: $\hat{e}(Q, P) = \hat{e}(H(h(m)), Y)$ or $\hat{e}(Q, P) = (\hat{e}(H(h(m)), Y))^{-1}$.

4 Floating Exponents

Let us stress the fact that splitting the secret exponent d from the RSA algorithm between the user and the SEM has additional benefits. If the RSA and inner signature [14] keys are broken, it is still possible to verify if a given signature was mediated by the SEM or not, provided that the later keeps a record of operations it performed. Should this verification fail, it becomes obvious that both keys have been broken and, in particular, the adversary was able to extract the secret exponent d. On the other hand, if the adversary wants to trick the SEM by offering it a valid partial RSASSA-PSS signature with a valid inner signature [14], he must know the right part d_u of the exponent d of the user whose keys he had broken. Doing this equals solving a discrete logarithm problem taken modulus each factor of N (though the factors length equals half of that of N). Therefore it is vital that no constraints, in particular on length, be placed on exponents d and their parts.

To mitigate the problem of smaller length of the factors of N, which allows solving the discrete logarithm problem with relatively small effort, a technique of switching exponent parts can be used. Let the SEM and the card share the same secret key K, which is unique for each card. After a signature is generated, the key deterministically

evolves on both sides. For each new signature, K is used as an initialization vector for a secure pseudo-random number generator (PRNG) to obtain a value that is added by the card to the part of the exponent it stores, and subtracted by the SEM from the part stored therein. This way, for each signature different exponents are used, but they still sum up to the same value. A one-time success at finding the discrete logarithm brings no advantage to the attacker as long as PRNG is strong and K remains secret.

To state the problem more formally, let K_i be a unique key shared by the card and sub-SEM_i, $i = 1, \ldots, t$ ($t \geq 1$). To generate an RSA signature the card does the exponentiation of the result of EMSA-PSS coding to the exponent equal to

$$d_u \pm \sum_{i=1}^{t} (-1)^i \cdot GEN(K_i), \tag{1}$$

where $GEN(K_i)$ is an integer output of a cryptographically safe PRNG (see e.g. generators in [28], *excluding* the Dual_EC_DRBG generator – for the reason see [29]). It suffices if length of $GEN(K_i)$ equals $\ell + \lfloor \log_2 N \rfloor + 1$, where ℓ is a fixed element from the set $\{128, \ldots, 160\}$. Operator "$\pm$" in Eq. (1) means that the exponent is alternately "increased" and "decreased" every second signature: this and multiplier $(-1)^i$ lessen changes of length of the exponent. Next, for each K_i the card performs a deterministic key evolution (sufficiently many steps of key evolution seem to be feasible on nowadays smart cards, cf. [30] claiming on p. 4, Sect. "E^{2}PROM Technology", even $5 \cdot 10^5$ write/erase cycles). To calculate its part of the signature, each sub-SEM_i exponentiates the result μ of EMSA-PSS coding (as received from the user along with the partial result of exponentiation) to the power of $d_{i,\mathrm{SEM}} \mp (-1)^i \cdot GEN(K_i)$. Next, the sub-$\mathrm{SEM}_i$ performs a deterministic evolution of the key K_i.

Note that should the card be cloned it will be revealed after the first generation of a signature by the clone – the SEM will make one key-evolution step further than the original card and the keys will not match. Each sub-SEM_i shall keep apart from its current state, the initial value of K_i to facilitate the process of investigation in case the keys get de-synchronized. To guarantee that the initial K_i will not be changed by sub-SEM_i, the following procedure might be applied: At point 2 of key generation procedure each sub-SEM commits to the initial K_i by broadcasting its hash $h(K_i)$ to other sub-SEMs. Next, at point 5 all broadcasted hashes are included in data set D, and are distributively signed by sub-SEMs with all the public data. Note that these hashes are sent to the card at point 7, and at points 7, 8 the card can check K_i against its commitment $h(K_i)$, $i = 1, \ldots, t$.

In order to force the adversary into tricking the SEM (i.e. make it even harder for him to generate a valid signature without participation of the SEM), one of the sub-SEMs may be required to place a timestamp under the documents (the timestamp would contain this sub-SEM's signature under the document and under the user's signature finalized by all the sub-SEMs) and only timestamped documents can be assumed valid. Such outer signature in the timestamp *must* be applied *both* to the document *and* to the finalized signature of the user. The best solution for it seems to be to use a scheme based on a completely different problem, to use a hash function signature scheme for instance. The Merkle tree traversal algorithm provides additional features with respect to timestamping: if a given sub-SEM faithfully follows the algorithm for any two document

signatures it is possible to reconstruct (based on the signature only, without an additional timestamp) the succession in which the documents have been signed. Note that other sub-SEMs will verify the outer hash-based signature as well as the tree traversal order.

If hash-based signatures are implemented in SEM, it is important to separate the source of randomness from implementation of the signatures (i.e. from key generation — apart from key generation this signature scheme is purely deterministic). Instead of one, at least two independent sources of randomness should be utilized and their outputs combined.

5 Forensic Analysis

As an example of forensic analysis consider the case of malicious behavior of one of the sub-SEMs. Suppose that the procedure of distributed RSA key generation bounds each sub-SEM$_i$ to its secret exponent $\tilde{d}_i$ (see point 4 of the ID-card's key generation procedure), for example by some checking signature made at the end of the internal procedure of generating the RSA key.

As we could see, the sub-SEM$_i$ cannot claim that the initial value of K_i was different than the one passed to the card. If correct elements $d_{i,\text{SEM}}$, K_i, $i = 1, \ldots, t$, were used in RSA signature generation at point 11 of the key generation procedure, and correct $d_{i,u}$ were passed to the ID-card, then the signature is valid. The sub-SEMs should then save all values $\beta_i = \mu^{\alpha_i} \bmod N$ generated by sub-SEM$_i$, $i = 1, \ldots, t$, to finalize the first cards partial signature s_u:

$$s = s_u \prod_{i=1}^{t} \mu^{\alpha_i} \bmod N.$$

Since $\alpha_i = d_{i,\text{SEM}} - (-1)^i \cdot GEN(K_i)$, and the initial value of K_i is bounded by $h(K_i)$, value β_i is a commitment of correct $d_{i,\text{SEM}}$.

Now consider the case of the first signature being invalid. First, the ID-card is checked: it reveals all values received: K_i, as well as received $d'_{i,u}$, $i = 1, \ldots, t$. Next, raising μ to power $(\sum_{i=1}^{t} d'_{i,u}) + \sum_{i=1}^{t} (-1)^i \cdot GEN(K_i)$ is repeated to check if partial signature s_u was correct. If it was, it is obvious that at least one sub-SEM behaved maliciously. All $\tilde{d}_i$ must be revealed, and integers $d'_{i,\text{SEM}} = \tilde{d}_i - d'_{i,u}$ are calculated. Having $d'_{i,\text{SEM}}$ and K_i it is easy to check correctness of each exponentiation $\mu^{\alpha_i} \bmod N$.

6 Implementation Recommendations

Hash Functions. Taking into account security aspects of long-term certificates used for digital signatures a hash function h used to make digests $h(m)$ should have long-term collision resistance. Therfore we propose to use the zipper hash construction [31], which utilizes two hash functions that are feed with the same message.

To harden the zipper hash against general techniques described in [32], we propose to use as the first hash function some non-iterative one, e.g. a hash function working analogously to MD6, when MD6's optional, mode control parameter L is greater than 27 (see Sect. 2.4.1 in [33]) – note that $L = 64$ by default.

RSA. It is advisable that modulus N of the RSA algorithm be a product of two *strong primes* [22]. Let us assume that the adversary succeeded in factorizing N into q and p. We do not want him to be able to gain any knowledge on the sum (1), that is indirectly on outputs of $GEN(K_i)$ for $i = 1, \ldots, t$. However, if $p-1$ or $q-1$ has a large smooth divisor, then by applying Pohling-Hellman algorithm he might be able to recover the value of sum (1) modulo the smooth divisor. Here "smooth" depends on adversary's computational power, but if p, q are of the form $2p' + 1$, $2q' + 1$, respectively, where p', q' are prime, then the smooth divisors for this case equal two only. Additionally, if the card and all the sub-SEM$_i$ unset the least significant bit of $GEN(K_i)$ then the output of the generator will not be visible in the subgroups of order two. In order to learn anything about (1), the adversary needs to perform an attack on discrete logarithm problem in the subgroup of large prime order (i.e. p' or q'). A single value does not bring much information and the same calculations must be carried out for many other intercepted signatures in order to launch cryptanalysis recovering keys K_i.

Elliptic Curves. The elliptic curve for the inner signature should have *embedding degree* ensuring at least 128-bit security (cf. [34]). Note that the security of the inner signature may not be entirely independent of the security of RSA – a progress made in attacks utilizing GNFS may have serious impact on *index calculations* (see last paragraph on p. 29 of online version [35]). Meanwhile, using pairing we need to take into account the fact, that the adversary may try to attack the discrete logarithm problem in the field in which verification of the inner signature takes place. Therefore we recommend a relatively high degree of security for the inner signature (see that according to Table 7.2 from [36], 128-bit security is achieved by RSA for 3248-bit modulus N, and such long N could distinctly slow down calculations done on the side of a smart card).

The proposed nested signature scheme with the zipper hash construction, extended with the secret keys shared between the card and sub-SEMs used for altering the exponent, and the SEM hash-signature under a timestamp, taken together increase the probability of outlasting the crypto analytical efforts of the (alleged) adversary. We hope that on each link (card $\rightarrow$ SEM, SEM $\rightarrow$ finalized signature with a timestamp) at least one out of three safeguards will last.

6.1 Resources and Logistics

If the computational and communication costs of distributed computation of strong RSA keys are prohibitively big to use this method on a large scale, one could consider the following alternative solution. Suppose there is a dealer who generates the RSA keys and splits each of them into parts that are distributed to the card and a number of sub-SEMs. When parts of the key are distributed, the dealer destroys its copy of the key.

Assume that the whole procedure of keys generation and secret exponents partition is deterministic, dependent of a random *seed* that is distributively generated by the dealer and sub-SEMs. For the purpose of verification for each key the parties must first commit to the shares of the *seed* they generated for that key. Next, some portion of the keys produced by the dealer as well as the partition of the secret exponents undergo a verification against commited shares of the *seed*. The verified values are destroyed afterwards.

The BLS key should be generated as described in Subsect. 3, necessarily *before* the RSA key is distributed.

Furthermore, each sub-SEM$_i$ generates its own secret key K_i to be used for altering the exponent, and sends it to the card (each sub-SEM$_i$ should generate K_i *before* it has obtained its part of the RSA exponent). One of the sub-SEMs or a separate entity designated for timestamping, generates its public key for timestamp signing (also *before* the RSA key is distributed). Note that this way there are components of the protocol beyond the influence of the trusted dealer (the same applies to each of the sub-SEMs).

Another issue are resources of the platform on which the system is implemented on signer's side. If the ID-card does not allow to generate the additional, inner signature efficiently, when the non-CRT implementation of RSA signatures must be executed, HMAC [37] function might be used as a source of a salt for the EMSA-PSS encoding. Let K_{MAC} be a key shared by the ID-card and *one* of the sub-SEM's, say sub-SEM$_j$. To generate a signature under message's digest $h(m)$, $salt = \text{HMAC}(h(m), K_{MAC})$ is calculated by the ID-card, and the signature generation on the user's side proceeds further as described above. On the SEM's side, after finalization of the RSA signature the EMSA-PSS encoding value μ is verified. The sub-SEM$_j$ possessing K_{MAC} can now check validity of *salt*. Note that K_{MAC} might evolve as keys K_i do, and K_{MAC} might be used instead of K_j (thus one key might be dropped from Eq. (1)). In case of key-evolution the initial value of K_{MAC} should also be stored by sub-SEM$_j$, to facilitate a possible investigation.

If BLS is replaced by HMAC, then a more space-efficient encoding function [38] may be used instead of EMSA-PSS. The scheme uses a single bit value produced by a pseudorandom number generator on the basis of a secret key (the value is duplicated by the encoding function). Thus this bit value might be calculated from HMAC($h(m)$, K_{MAC}). Note that also in this case the evolution of K_{MAC} is enough to detect the fact that ID-card has been cloned, even if other keys K_i from (1) are not used in the system: usually a pseudorandom sequence and its shift differ every few possitions.

Yet another aspect that influences the system is the problem of trusted communication channels between the dealer and the card, and between each sub-SEM and the card. If these are cryptographic (remote) channels, then, above all, *security of the whole system will depend on the security of the cipher in use*. Moreover, if a public-key cipher is to be used, the question remains as to who is going to generate the public key (and the corresponding *secret* key) of the card? It *should not* be the card itself, neither its manufacturer. If, on the other hand, a symmetric cipher was used, then how to deliver the key to the card remains an open question. A distinct symmetric key is needed on the card for each sub-SEM and, possibly, for the dealer.

Therefore (above all, in order to eliminate the dependence of the signing schemes from the cipher scheme(s)), the best solution would be to transfer the secret data into the card directly on site where the data is generated (i.e. at the possible dealer and all the subsequent sub-SEMs). Such a solution can have its influence on the physical location of sub-SEMs and/or means of transportation of the cards.

Final Remarks

In this paper we have shown that a number of practical threats for PKI infrastructures can be avoided. In this way we can address most of the technical and legal challenges

for proof value of electronic signatures. Moreover, our solutions are obtained by cryptographic means, so they are independent from hardware security mechanisms, which are hard to evaluate by parties having no sufficient technical insight. In contrast, our cryptographic solutions against hardware problems are platform independent and self-evident.

References

1. Young, A., Yung, M.: The dark side of "Black-box" cryptography, or: Should we trust capstone? In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 89–103. Springer, Heidelberg (1996)
2. Young, A., Yung, M.: The prevalence of kleptographic attacks on discrete-log based cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 264–276. Springer, Heidelberg (1997)
3. Young, A.L., Yung, M.: A timing-resistant elliptic curve backdoor in RSA. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) Inscrypt 2007. LNCS, vol. 4990, pp. 427–441. Springer, Heidelberg (2008)
4. Young, A., Yung, M.: A space efficient backdoor in RSA and its applications. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 128–143. Springer, Heidelberg (2006)
5. Young, A., Yung, M.: An elliptic curve backdoor algorithm for RSASSA. In: Camenisch, J.L., Collberg, C.S., Johnson, N.F., Sallee, P. (eds.) IH 2006. LNCS, vol. 4437, pp. 355–374. Springer, Heidelberg (2007)
6. Boneh, D., Ding, X., Tsudik, G., Wong, C.M.: A method for fast revocation of public key certificates and security capabilities. In: SSYM 2001: Proceedings of the 10th Conference on USENIX Security Symposium, p. 22. USENIX Association, Berkeley (2001)
7. Tsudik, G.: Weak forward security in mediated RSA. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 45–54. Springer, Heidelberg (2003)
8. Boneh, D., Ding, X., Tsudik, G.: Fine-grained control of security capabilities. ACM Trans. Internet Techn. 4(1), 60–82 (2004)
9. Bellare, M., Sandhu, R.: The security of practical two-party RSA signature schemes. Cryptology ePrint Archive, Report 2001/060 (2001)
10. Coppersmith, D., Coron, J.S., Grieu, F., Halevi, S., Jutla, C.S., Naccache, D., Stern, J.P.: Cryptanalysis of ISO/IEC 9796-1. J. Cryptology 21(1), 27–51 (2008)
11. Coron, J.S., Naccache, D., Tibouchi, M., Weinmann, R.P.: Practical cryptanalysis of ISO/IEC 9796-2 and EMV signatures. Cryptology ePrint Archive, Report 2009/203 (2009)
12. RSA Laboratories: PKCS#1 v2.1 — RSA Cryptography Standard + Errata (2005)
13. Jonsson, J.: Security proofs for the RSA-PSS signature scheme and its variants. Cryptology ePrint Archive, Report 2001/053 (2001)
14. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. J. Cryptology 17(4), 297–319 (2004)
15. Zhang, F., Safavi-Naini, R., Susilo, W.: An efficient signature scheme from bilinear pairings and its applications. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 277–290. Springer, Heidelberg (2004)
16. Buchmann, J., Dahmen, E., Klintsevich, E., Okeya, K., Vuillaume, C.: Merkle signatures with virtually unlimited signature capacity. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 31–45. Springer, Heidelberg (2007)
17. Kubiak, P., Kutyłowski, M., Lauks-Dutka, A., Tabor, M.: Mediated signatures - towards undeniability of digital data in technical and legal framework. In: 3rd Workshop on Legal Informatics and Legal Information Technology (LIT 2010). LNBIP. Springer, Heidelberg (2010)
18. Boneh, D., Franklin, M.: Efficient generation of shared RSA keys. J. ACM 48(4), 702–722 (2001)

19. Malkin, M., Wu, T.D., Boneh, D.: Experimenting with shared generation of RSA keys. In: NDSS. The Internet Society, San Diego (1999)
20. Frankel, Y., MacKenzie, P.D., Yung, M.: Robust efficient distributed RSA-key generation. In: PODC, vol. 320 (1998)
21. Gilboa, N.: Two party RSA key generation (Extended abstract). In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 116–129. Springer, Heidelberg (1999)
22. Algesheimer, J., Camenisch, J., Shoup, V.: Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. Cryptology ePrint Archive, Report 2002/029 (2002)
23. MacKenzie, P.D., Reiter, M.K.: Delegation of cryptographic servers for capture-resilient devices. Distributed Computing 16(4), 307–327 (2003)
24. Coron, J.S., Icart, T.: An indifferentiable hash function into elliptic curves. Cryptology ePrint Archive, Report 2009/340 (2009)
25. Coron, J.-S.: On the Exact Security of Full Domain Hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (2000)
26. Coron, J.-S., Joux, A., Kizhvatov, I., Naccache, D., Paillier, P.: Fault attacks on RSA signatures with partially unknown messages. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 444–456. Springer, Heidelberg (2009)
27. Coron, J.-S., Naccache, D., Tibouchi, M.: Fault attacks against EMV signatures. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 208–220. Springer, Heidelberg (2010)
28. Barker, E., Kelsey, J.: Recommendation for random number generation using deterministic random bit generators (revised). NIST Special Publication 800-90 (2007)
29. Shumow, D., Ferguson, N.: On the possibility of a back door in the NIST SP800-90 Dual EC Prng (2007), http://rump2007.cr.yp.to/15-shumow.pdf
30. Infineon Technologies AG: Chip Card & Security: SLE 66CLX800PE(M) Family, 8/16-Bit High Security Dual Interface Controller For Contact based and Contactless Applications (2009)
31. Liskov, M.: Constructing an ideal hash function from weak ideal compression functions. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 358–375. Springer, Heidelberg (2007)
32. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
33. Rivest, R.L., Agre, B., Bailey, D.V., Crutchfield, C., Dodis, Y., Elliott, K., Khan, F.A., Krishnamurthy, J., Lin, Y., Reyzin, L., Shen, E., Sukha, J., Sutherland, D., Tromer, E., Yin, Y.L.: The MD6 hash function. a proposal to NIST for SHA-3 (2009)
34. Granger, R., Page, D.L., Smart, N.P.: High security pairing-based cryptography revisited. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 480–494. Springer, Heidelberg (2006)
35. Lenstra, A.K.: Key lengths. In: The Handbook of Information Security, vol. 2, Wiley, Chichester (2005), http://www.keylength.com/biblio/Handbook_of_Information_Security_-_Keylength.pdf
36. Babbage, S., Catalano, D., Cid, C., de Weger, B., Dunkelman, O., Gehrmann, C., Granboulan, L., Lange, T., Lenstra, A., Mitchell, C., Näslund, M., Nguyen, P., Paar, C., Paterson, K., Pelzl, J., Pornin, T., Preneel, B., Rechberger, C., Rijmen, V., Robshaw, M., Rupp, A., Schläffer, M., Vaudenay, S., Ward, M.: ECRYPT2 yearly report on algorithms and keysizes (2008-2009) (2009)
37. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (Informational) (1997)
38. Qian, H., Li, Z.-b., Chen, Z.-j., Yang, S.: A practical optimal padding for signature schemes. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 112–128. Springer, Heidelberg (2006)