New generation of secure and practical RSA-based signatures

R.J.F. Cramer

Computer Science/Department of Algorithmics and Architecture

CS-R9557 1995

Report CS-R9557
ISSN 0169-118X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

New Generation of Secure and Practical RSA-based Signatures

Ronald Cramer

CWI

P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract

For most digital signature schemes used in practice, such as ISO9796/RSA or DSA, it has only been shown that certain plausible cryptographic assumptions, such as the difficulty of factoring integers, computing discrete logarithms or the collision-intractability of certain hash-functions are necessary for the security of the scheme, while their sufficiency is, strictly speaking, an open question.

A clear advantage of such schemes over many signature schemes with security proven relative to such common cryptographic assumptions, is their efficiency: as a result of their relatively weak requirements regarding computation, bandwidth and storage, these schemes have so far beaten proven secure schemes in practice.

Our aim is to contribute to the bridging of the gap that seems to exist between the theory and practice of digital signature schemes. We present a digital signature that offers *both* proven security and practical value.

More precisely, under an appropriate assumption about RSA, the scheme is proven to be not existentially forgeable under adaptively chosen message attacks. Furthermore, we identify some electronic devices where our scheme can be conveniently implemented using dedicated smartcards that are available today.

AMS Subject Classification (1991): 94A60
CR Subject Classification (1991): D.4.6
Keywords & Phrases: Cryptography, Security, Digital Signatures, RSA.

1. Introduction

Consider, very generally, electronic transaction systems that require message authentication mechanisms such as digital signature schemes. Although we do not mean to limit ourselves to this case in this paper, assume that the individual players have dedicated (i.e., capable of performing public key cryptography) smartcards as available today or in the near future, as their user-devices. We will simply say that a digital signature scheme has practical value in this context, if it can be conveniently used, given the available infrastructure *and* hardware.

Our objective is to design a digital signature scheme that offers *both* high security and practical value. Informally, we require the following of our target scheme. Fist, relative to some plausible cryptographic assumption, a proof must be given that the scheme is not existentially forgeable under adaptively chosen message attacks [3]. Without attempting

E-mail address: `cramer@cwi.nl`.

to quantify the efficiency needed, we require, secondly, that the amount of computation and the size of the signatures are small, and, finally, that the amount of storage needed is reasonably limited.

In a sequence of results [12], [4], [13] and finally [14], it was established that the existence of one-way functions is necessary and sufficient for the existence secure signatures. This result, however theoretically very important, does not give rise to a practical signature scheme. The construction, which is based on a general *one-way function*, uses a costly "bit-by-bit" signing technique in conjunction with tree authentication [12]. As a result, the size of signatures is $O(k^2 \cdot \log i)$, where k stands for a security parameter and i indicates the number of signatures made.

Benefitting from the special properties of *claw-free trapdoor permutations*, the secure scheme presented in [3] achieves signatures of size $O(k \cdot \log i)$ instead. Their scheme also uses a tree structure. Intractibility of factoring is a sufficient assumption for the existence of the family of functions required for their scheme (for a generalization, see [9]. Though yielding shorter signatures asymptotically, the size grows rapidly in practice as the number of signatures made increases.

Starting with [2], many practical digital signature schemes have been proposed, for instance, [15], [16], [17], [18], [19], [20] and [21]. Although many of them are actually used in practice today, these schemes seem to have the property that their security is hard to analyze. We certainly do not mean to suggest here that their security is dubious. On the contrary, these schemes rely on common cryptographic assumptions, such as the difficulty of factoring, the difficulty of computing discrete logarithms or the collision intractability of certain hash functions, and have so far resisted many years of cryptanalytic efforts.

However, none of these schemes has been shown to be secure in the sense of [3] provided that any of these mentioned cryptographic assumptions holds. This implies that, independently of their validity, these necessary and common cryptographic assumptions may still turn out to be insufficient for the security of these signature schemes. Thus, based on the above, it is still an open problem to design a secure and truly practical digital signature scheme, that may be used in today's or tomorrow's information systems.

Recently, progress has been made in this area. By the work of [8], and subsequently [10], it can be concluded that the first two requirements, namely proven security, moderate amount of computation and provision of any reasonable number of small-sized signatures, can be satisfied. The cryptographic assumptions needed there, are an RSA-assumption and the factoring assumption (or more precisely, the existence of a particular family of claw-free trapdoor permutations), respectively. For efficient *fail-stop signatures*, see [7]. These schemes yield practically much smaller signatures compared to, for instance, [3]. The reason is that, instead of binary authentication trees, these schemes allow the use of trees with much larger branching degree.

Briefly, the efficiency of these schemes is as follows. Let integers l, d and a security parameter k be given (in [8] it must be required that $l \geq k$). In both [8] and [10], a signer can make at least l^d signatures. The size of a signature in [8] amounts to dk bits, whereas a signature in [10] has size $(3d-2)k$ bits [1]. The main contribution of [10] resides in the fact that it is based on a general and potentially weaker cryptographic assumption. In both schemes, the idea is to choose l large, such that for any reasonable number of signatures the resulting size of the signatures is small.

Theoretically, these schemes offer a trade-off, via the flexibility of choosing l, between the size of signatures and the storage required: the size is $O(\frac{k}{\log l} \cdot \log i)$ bits, with $O(l \cdot k)$ bits storage for the system constant and $O(\frac{k}{\log l} \cdot \log i)$ bits dynamic storage for the signer. The corresponding figures for [3] and [9], are $O(k \cdot \log i)$ bits for the size of signatures and $O(k \cdot \log i)$ bits storage [2]. Both schemes [8] and [10], however, have the disadvantage that all signers and receivers of signatures must have access to a large list of random numbers. In [8] and [10], this lists consists of l random k-bit strings. Furthermore, [8] requires that l primes are also given.

In [8], authentication of computer faxes is identified as an application where their proposed scheme is certainly useful (this equally applies to the scheme of [10]). However, in any practical system that uses smartcards as the main players, this assumption about shared access to the list of random numbers seems unreasonable, simply because of its storage requirements (in case a user has a *wallet with observer* ([22], [23]) as user device, there are solutions, though not as efficient as the scheme presented in this paper, that preserve the off-line property). One can envision a system where the players gain access to the list through a server. If this server and the communication link are trusted, this solution has only the on-line character as the main disadvantage. Otherwise, one also has to employ mechanisms for ensuring the integrity of the supplied data (one-way accumulators [25] seem to allow for an efficient approach).

Our contribution is the design of a secure signature scheme where the size of the signatures is $(d+1)k$ bits, while l^d signatures can be made. The storage for the signer is approximately $(d+3)k + l \log k$ bits, thus improving considerably over [8] and [10]. The integers l and d can be chosen independently from the security parameter k. The security is derived from an appropriate RSA-assumption.

Thus, asymptotically speaking, size $O(\frac{k}{\log l} \cdot \log i)$ bits for the size of signatures is retained compared to [8] and [10], while the storage is cut down to $O(k + l \cdot \log k)$ bits.

For estimating the practical value of this result, take, for instance, $k = 1000$, which corresponds to an RSA-modulus of size 1000 bits, $l = 1000$ and $d = 3$. One readily sees that over a billion signatures have size at most 4000 bits each, while the storage for the signer is approximately 2 Kbytes, including 1.5 Kbytes for a "compressed" list of 1001 primes. The latter list, which is a system constant, must also be accessible by any receiver of a signature from our scheme. For generation and verification of a signature, only a few

[1]The actual sizes stated in [8] and [10] are larger. However, these can be reduced by roughly a factor of two if one observes that the signatures are redundant, i.e., part of the signature can be recalculated from another part. See also Section 3.

[2]The dynamic storage can be reduced by applying a suggestion from [5].

RSA-exponentiations are required. This example indicates that our secure scheme may very well be implemented in a system that uses today's smartcards. See also Section 7.

Our exposition is organized as follows. In Section 2, we outline the technical ideas behind our design. The formal presentation of the scheme can be found in Section 3. The latter section left open the choice of some parameters. This is resolved in Section 4, which is followed by a discussion of the performance of our scheme in Section 5. The proof of security is given in Section 6. We conclude our presentation with applications where implementation of our scheme might be desirable in Section 7.

2. Basic Idea

Conceptually, our signature scheme may be viewed as a cross between [3] and [8], together with modifications enabling their synthesis. In [8], all players in the signature scheme must have access to two lists. The first list contains l primes. Depending on the particular RSA-assumption one wishes to make, these could be, for instance, the first l uneven primes, or l random primes. The second list consists of l random k-bit strings. Our first objective is to remove the necessity of the list of random numbers.

In [8], the system constants are as follows. Let L denote the list of primes $\{p_0, \ldots, p_{l-1}\}$, L' the list of l random l-bit strings $\{x_0, \ldots, x_{l-1}\}$ and let a denote a random l-bit string, to be used as the root of all authentication trees. Let a signer be given an RSA-modulus n together with its factorization.

The "basic authentication step" in [8] is

$$y \leftarrow (\alpha \cdot \prod_{i=0}^{l-1} x_i^{\beta_i})^{\frac{1}{p_j}} \bmod n$$

where α is an already authenticated value, $\beta = \beta_0 || \cdots || \beta_{l-1}$ is an l-bit string to be authenticated, and p_j is a prime from the list L that has not been used before in connection with α.

Instead, our basic authentication step is

$$y \leftarrow (\alpha \cdot h^{\beta})^{\frac{1}{p_j}} \bmod n,$$

where h is a member of $\mathbb{Z}_n^*$ and part of the signer's public key. Here the values that can be authenticated are elements of $\mathbb{Z}_n^*$. This implies that the primes p_j must be larger than n. If n is a k bits modulus, we will take the p_j to be of size $k+1$ bits. This removes the list L'. If we now take L to be consisting of l consecutive $k+1$ bit primes, and store only the first prime and all consecutive differences, the storage of the list is approximately $l + l \log k$ bits, if we assume the differences between consecutive primes to have expected size $\log k$ bits [24].

However, implementing this idea only results in a scheme that we can prove secure against random message attacks. Such a scheme can be efficiently transformed to a scheme that is secure against active attacks, as is desired here, by means of a technique described in [11]. The loss of efficiency is a factor of two (twice as much computation, signature size twice as large). But we can do better in this case, if we add one prime q $(q > n)$ with a special purpose to the list: it is only used when a message m, agreed upon between the signer and a receiver, is to be authenticated, as follows

$$z \leftarrow (\alpha \cdot h^m)^{\frac{1}{q}} \bmod n,$$

where α is a "freshly" generated leaf in the authentication tree. This relates to the idea [3] of applying sufficiently independent functions to the actual signing process and the construction of an authentication tree, respectively.

3. Description of the Scheme

In a preprocessing-phase, a security parameter k is determined, as well as integers l and d. Next, a list L consisting of $l+1$ distinct primes is generated. These primes all have length (at least) $k+1$ bits. Say $L = \{q, p_0, \ldots, p_{l-1}\}$. Ways of choosing L are discussed in the next section.

Furthermore, we assume that we are given a probabilistic polynomial time generator G that, on input 1^k, outputs a triple (n, r, s), where r and s are primes and $n = r \cdot s$ is a k bits integer. It is assumed that G is defined such that it is infeasible to factor n, when only n as generated by G is given as input. A typical implementation of G is as follows. On input 1^k, select two primes r and s, of size approximately $\frac{k}{2}$ bits, independently at randomand compute their product n. Finally, output (n, r, s). If any of the primes in the list L is not co-prime with $(r-1)(s-1)$, G must be invoked again, until n is such that all exponents are indeed RSA-exponents.

The algorithm DFS(i), which is used in the formal description of our scheme, gradually develops a full l-ary tree of depth d, where the nodes are selected at random from $\mathbb{Z}_n^*$. The tree is constructed in depth-first fashion. Although not explicitly given as input to DFS(i), it is assumed that it has access to l, d, x_0 and n. The value x_0 serves as the root of the tree. Each time DFS(i) is invoked $(i = 1 \ldots l^d)$, it creates a path to a new leaf x_d and outputs this path, say, $x_1, \ldots, x_d$ (the root x_0 being understood). This sequence is ordered such that x_{j-1} is the parent of x_j $(j = 1 \ldots d)$.

Furthermore, for each node x_j in this sequence, DFS(i) also outpus an indicator i_j $(j = 1 \ldots d)$ in such a way that i_j is assigned to x_j if and only if x_j is the i_j-th child of x_{j-1}. The amount of storage needed for this procedure (apart from l, d, x_0 and n) does not exceed the amount of storage needed for $d-1$ pairs consisting of a node and an indicator.

We start with an informal overview of the scheme. The signer has as public key an RSA-modulus n, $h \in \mathbb{Z}_n^*$ and $x_0 \in \mathbb{Z}_n^*$. As always, his knowledge of the factorization (r, s) of n enables him to compute $X^{\frac{1}{v}} \bmod n$ for any $X \in \mathbb{Z}_n^*$ and any integer v such that $\gcd(v, (r-1)(s-1)) = 1$.

The signer gradually constructs, in a depth first fashion, an l-ary authentication tree with depth d: each time a new signature is required he constructs a path to a new leaf. All nodes x are members of $\mathbb{Z}_n^*$, given by their smallest non-negative representative modulo n. The message space is equal to the set $\{0,1\}^k$, which we will also identify with the set of non-negative integers smaller than 2^k. Note that by the choice of the size of the primes in the list L, the modulus n is smaller than each of these primes.

In Figure 1, the signer is making his i-signature, on a message $m \in \mathbb{Z}_n^*$. So, in particular x_d is the i-th leaf he reaches. The part of the tree on the right side of the path $x_0, \ldots, x_{d-1}, x_d$ is not yet constructed. Since x_1 happens to be the i_1-st child of x_0, the signer authenticates x_1 with respect to the prime p_{i_1} by computing $y_1 \leftarrow (x_0 \cdot h^{x_1})^{\frac{1}{p_{i_1}}} \bmod n$. Similar rules apply to the authentication of the remaining nodes in this path. In particular, it so happens to be in our example that x_d is the i_d-th child of x_{d-1}. Thus x_d is authenticated by computing $y_d \leftarrow (x_{d-1} \cdot h^{x_d})^{\frac{1}{p_{i_d}}} \bmod n$. Finally, the message m is signed by computing $z \leftarrow (x_d \cdot h^m)^{\frac{1}{q}} \bmod n$. Notice that the prime q is only used when the "actual signature" is computed, while the other primes in the list L are used exclusively in the process of constructing the authentication tree. The signature on m consists of the y_j and indicators i_j, $(j = 1 \ldots d)$ and z.

Concerning the storage needed for the signer, notice that the part of the tree left from the path $(x_0, \ldots, x_{d-1}, x_d)$ can be deleted. Actually, x_d itself can be removed. In order to carry on with the depth-first construction of the tree, it is sufficient to store $x_0, \ldots, x_{d-1}$ and the indicators to their parents. This storage amounts to at most $(d-1)(k+\log l)$ bits (the root x_0 is part of the public key).

A receiver of this signature gets only the message m, authentication values y_j, the indicators i_j $(j = 1 \ldots d)$ and z. So, what about the nodes? These are re-computed as follows. On input of the public key, the list L, m and z he recomputes x_d as $x_d \leftarrow z^q \cdot h^{-m} \bmod n$. Recursively, the receiver re-computes x_{j-1} from x_j, y_j and i_j in a similar fashion $(j = d \ldots 1)$. The last node x_0 he thus computes should be equal to the actual x_0, which is part of the public key. If so, the signature is accepted. We point out that in many tree-structured signature schemes, it is sufficient to send the authentication values and have the verifier re-compute the nodes, instead of defining these as part of the signature. It is easily seen why this does not effect the security at all: briefly, if the verifications in the "reduced" scheme hold, one gets a signature in the original scheme (on the same message, of course) by simply incorporating the re-computed nodes. The remark in a footnote in Section 1 is based on this observation.

More formally, the description of the new signature scheme is as follows.

Preprocessing:
A security parameter k, integers l and d are determined. A list $L = \{q, p_0, \ldots, p_{l-1}\}$ consisting of $l+1$ primes of size $k+1$ bits is generated (See Sections 4 and

Initialization:
The signer runs $G(1^k)$ and obtains a triple (n, r, s). Next, he chooses h and x_0 at random in $\mathbb{Z}_n^*$. His public key pk is now the pair (n, h, x_0), while his secret key sk

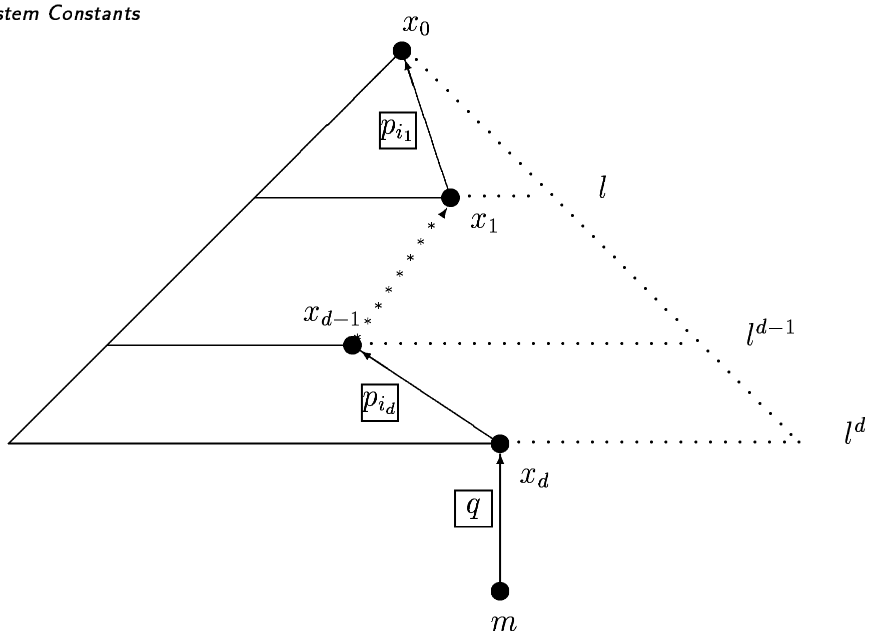

Figure 1: The i-th Signature

consists of the pair (r, s).

Signing:
Let a k bit message m be given. Then the i-th signature, where $1 \leq i \leq l^d$, is computed as follows. First, the signer puts $(x_1, i_1, \ldots, x_d, i_d) \leftarrow \mathrm{DFS}(i)$. Next, he computes (for $j = 1 \ldots d$) $y_j \leftarrow (x_{j-1} \cdot h^{x_j})^{\frac{1}{p_{i_j}}} \bmod n$. Finally, he computes $z \leftarrow (x_d \cdot h^m)^{\frac{1}{q}} \bmod n$. The signature σ on m consists of the values $z, y_1, i_1, \ldots, y_d, i_d$.

Verification: Verification is done as follows. The receiver of a signature puts $\sigma = (Z, Y_1, i_1 \ldots, Y_d, i_d)$, and, on input of $pk = (n, h, x_0)$, m and σ, he computes $X_d \leftarrow z^q \cdot h^{-m} \bmod n$. Finally, he computes $X_{j-1} \leftarrow Y_j^{\frac{1}{p_{i_j}}} \cdot h^{-X_j} \bmod n$ $(j = d \ldots 1)$. If $X_0 = x_0 \bmod n$, the signature is accepted.

Remark 1 *For convenient exposition of the scheme, we have chosen to let the signer only use the leaves for signing. However, the scheme is easily adapted so as to allow for a more extensive use of the authentication tree. In this modified scheme, each freshly constructed node can immediately be used for making a signature. The proof of security is easily adapted to fit with this modification.*

4. Generating the System Constants

In order to mimimize the storage needed for the system constants, i.e., the list L consisting of $l+1$ distinct $k+1$ bit prime numbers, k, l and d, it is convenient to set L to any $l+1$

consecutive $k+1$ bit primes. From the prime number theorem, the expected size of L is about $k + l \cdot \log k$ bits for large enough k, if only the first prime and the differences between consecutive primes are stored. The generation of L can be performed in probabilistic polynomial time.

It must be stressed, however, that the correctness of the scheme is independent of the particular ways of generating L. Also, the proof of security is not affected by such choices. What is dependent on the choice of L, is the particular assumption we have to make about RSA-inversion. See Section 6.

5. Performance of the Scheme

A signer can make at least l^d signatures (see also Remark 1) such that the size of each signature does not exceed $(d+1)k$ bits. A public key has size $3k$ bits. In a possible variation of the scheme, the values of h and x_0 are chosen, mutually at random, as k-bit strings and are the same for each signer (see also [8]).

Concerning the amount of computation needed, signing only requires one full RSA-exponentiation and one modular multiplication on-line. A path to the current leaf can be authenticated by pre-processing, using d full-RSA exponentiations and d modular multiplications. A receiver of a signature will have to perform $d+1$ full RSA-exponentiations and $d+1$ modular multiplications.

Both signer and receiver must have access to the list L. If this list is generated as suggested in Section 4, the storage needed is approximately $k + l \cdot \log k$ bits. For the gradual depth-first construction of the authentication tree, the signer stores at most $(d-1)(k + \log l)$ bits at any time. Secure storage in the strongest sense (i.e., storage not accessible or alterable by "the outside world") is only needed for the secret key (k bits) and the relevant nodes of the latest path in the tree, which is at most $(d-1)k$ bits. The public list L only has to be securely stored in a weaker sense: the signer must have certainty that L is authentic.

6. Proof of Security

The proof of security works for choice of the list L. However, the particular assumption we make about the difficulty of RSA-inversion depends on this very choice in the following way. If we have an algorithm $H(1^k, 1^l)$ that generates the list L, the assumption we require is the following.

Let be of polynomial size in k. Suppose we are given an RSA-modulus n as generated by $G(1^k)$ and a list L as generated by $H(1^k, 1^l)$ and let x be any member of $\mathbb{Z}_n^*$. Then there is no algorithm that has non-negligible probability of computing $x^{\frac{1}{v}} \bmod n$ with $v \in L$, on input G. H, n, l and x.

In the assumption below, we make a particular choice that minimizes the storage of L, namely of having L consist of $l+1$ consecutive primes. Furthermore, for reasons of simplicity, we require that these are the first $l+1$ primes of size $k+1$ bits.

Assumption 1 *Let L be as above. Let n be an RSA-modulus as generated by $G(1^k)$ and let x be a random member of $\mathbb{Z}_n^*$. Then there is no algorithm that has non-negligible probability of computing $x^{\frac{1}{v}} \bmod n$ with $v \in L$, on input L, n and x.*

Under this assumption, we can prove that the signature scheme is not existentially forgeable under adaptively chosen message attacks.

We are given integers l and d, a list $L = \{q, p_0, \ldots, p_{l-1}\}$ consisting of $l+1$ primes of size $k+1$ bits, and an RSA-modulus n such that $\phi(n)$ is co-prime with each of the primes in L. We assume that n is generated according to $G(1^k)$. Let $v \in L$.

In the following we show that we can set up a "simulated" signer, who gets as input $h \in \mathbb{Z}_n^*$ and $h^{\frac{1}{u}} \bmod n$ for all u in L different from v, but is yet indistinguishable from a signer who proceeds as in Section 3 after he is given h, n *and* its factorization. To this end, we consider two cases separately and focus mainly on the differences with Section 3.

In the first case $v = q$, the root x_0 is computed as $x_0 \leftarrow a_0^{p_0 \cdots p_{l-1}} \bmod n$, for randomly chosen a_0 from $\mathbb{Z}_n^*$. The value a_0 is stored. All nodes x, excluding the leaves, are computed as $x \leftarrow a^{p_0 \cdots p_{l-1}} \bmod n$, where a is chosen at random from $\mathbb{Z}_n^*$. The value a is stored. If any x is the e-th child of his parent $x_* = a_*^{p_0 \cdots p_{l-1}} \bmod n$, the authentication value y is computed as $y \leftarrow a_*^{p_0 \cdots p_{e-1} \cdots p_{e+1} \cdots p_{l-1}} \cdot (h^{\frac{1}{p_e}})^x \bmod n$. With the i-th signature, on a message m, the i-th leaf x is computed as $x \leftarrow a^q \cdot h^{-m} \bmod n$ where a is chosen at random from $\mathbb{Z}_n^*$. With the i-th signature request, the simulated signer can reveal the path to the i-leaf, together with all authentication values, and the authentication value $z = a$ of the message m.

In case $v \neq q$, say, $v = p_j$, the authentication tree has to be constructed from the bottom up. We first show how this is done for $d = 1$. Let x be the value that is to be the j-th child. The parent x_* is then computed as $x_* \leftarrow b^{p_0 \cdots p_{l-1}} h^{-x} \bmod n$, where b is chosen at random from $\mathbb{Z}_n^*$. The value b is stored. The authentication value y of x is computed as $y \leftarrow b^{p_0 \cdots p_{j-1} p_{j+1} \cdots p_{l-1}} \bmod N$. If x' is a different leaf from x, say it is the f-th child of x_*, its authentication value y is computed as $y \leftarrow b^{p_0 \cdots p_{f-1} p_{f+1} \cdots p_{l-1}} \cdot (h^{\frac{1}{p_f}})^{x'-x} \bmod N$. When we have constructed $l-1$ other such trees with $d = 1$, the same procedure can be used to combine them into a tree with $d = 2$, by letting the roots play the role of the leaves as above. By induction, we can build an l-ary tree with any depth d. One choice has been left open so far. The leaves x of the target tree of depth d must be chosen as $x \leftarrow b^q \bmod n$, for random b in $\mathbb{Z}_n^*$. With the i-th signature request, the simulated signer can reveal the path to the i-leaf, together with all authentication values, and the authentication value $z \leftarrow b \cdot (h^{\frac{1}{q}})^{-m} \bmod n$.

It is clear that in both cases each node in the tree has the uniform distribution and is independent of anything else. All other values follow determistically. Thus this simulation cannot be distinguished from the real signer.

The next step in our proof is to show that the existence of a successfull attacker, who is allowed to mount an adaptively chosen message attacks against the signature scheme, implies the existence of an algorithm that gets as input a modulus n (but not its factorization) as generated by G, the list L and any $x \in \mathbb{Z}_n^*$, and will output $x^{\frac{1}{v}} \bmod n$, for some $v \in L$, with essentially the same success probability.

This works as follows. We choose a random $v \in L$, and select any $x \in \mathbb{Z}_n^*$, and ρ at random from $\mathbb{Z}_n^*$. Put $h \leftarrow x^{\prod_{w \in L/\{v\}} w} \cdot \rho^{\prod_{w \in L} w} \bmod n$. Next we feed L, n, h, and $h^{\frac{1}{w}} \bmod n$ for all w in L different from v to the simulated signer and run the simulation. Also we run the attacker against this simulator. Assume that after l^d calls to the simulated signer, the attacker outputs a forgery, i.e., a signature on a message $\tilde{m}$ that has not been signed by the simulator. It is easily derived that this forgery must hook into the authentication tree in such a way that the simulator has presented a signature from which a node x, a prime $u \in L$, a value α and $(x \cdot h^{\alpha})^{\frac{1}{u}} \bmod n$, can be derived, while from the forgery the same node x, the same prime $u \in L$, a value $\tilde{\alpha}$ and $(x \cdot h^{\tilde{\alpha}})^{\frac{1}{u}} \bmod n$ but with $\tilde{\alpha} \neq \alpha \bmod n$ can be derived. It follows immediately that from these data, one can easily compute $h^{\frac{1}{v}} \bmod n$. From the perfectness of the simulation the probability that $u = v$ is at least $\frac{1}{2l}$. In this case we can also easily compute $x^{\frac{1}{v}}$ from $h^{\frac{1}{v}} \bmod n$. Thus, if the attacker has non-negligible succes probability, than we can extract v-th roots also with non-negligible probability, for some $v \in L$. This proves the following theorem.

Theorem 1 *Under Assumption 1, the signature scheme presented in Section 3 is not existentially forgeable under adaptively chosen message attacks.*

7. Applications

We investigate some applications of our scheme in systems that require a digital signature scheme that offers both efficiency and high security Here we first discuss a system where individuals have a tamperresistant smartcard with crypto-coprocessor as user-device. Let us assume that part of the functionality of the card is devoted to producing digital signatures on behalf of its holder. The receiving party may also be represented by a similar card.

If we assume that the signer is not likely to make more than one million signatures in the lifetime of the card, which may be reasonable in many applications, one could for instance set the parameters as follows. Take $k = 1000$, $l = 100$ and $d = 3$. Total storage for the signer does not exceed 1 Kbyte at any time, reserving $\log_2 1000$ bits for the differences between two consecutive primes in the list L. The receiver must have access to approximately 0.25 Kb (i.e., the list L, which is a system constant). Out of this 1 Kbyte, 3000 bits have to be stored securely (inaccessible and unalterable by the outside world). The size of each signature (up to one million signatures) is not larger than 4000 bits. Verifying a signature, for instance, requires 4 full RSA-exponentiations and 4 modular multiplications. A scenario where people log in to their workstation and insert their private smartcard whenever they wish or are required to sign data they send over Internet, can easily be based on the signature scheme presented here. The card can fully operate on its own and does not require communication with any server.

In general, selecting the security parameter k and the values of l and d, is a task that strongly depends on the particular application the scheme is part of. In any case one has the flexibility of choosing l, d and the secuirty k indepently from one another. This means that one can trade-off between size of signatures and amount of computation on one hand and the size of the system parameter on the other hand.

A more specific application is message authentication over Internet, possibly as part of electronic commerce services requiring high security digital signatures. If these applications need a larger signature bound than 1 million, one could set $l = 1000$ and $d = 3$ for instance, and be able to make over a billion signatures with size at most 4000 bits.

Note that the schemes from [8] and [10] also make sense in this scenario. However, as remarked before, they need secure access (i.e., they need to be sure about the athenticity) to a list of data that seems to be too large for storage on a smartcard, and can as such not be operated in an off-line fashion.

Yet another known application concerns the use of digital passports. One could imigine that both high security and efficiency are important feautures of any system implementing digital passports. The idea is very simple. An individual carries an ordinary smartcard, or a memory card, containing a digitalized passport, including a digitalized photograph of the holder. The card also stores a digital signature on the data that constitute the passport. This signature is provided by the government of the holder's home-country.

If the United States, for instance, were to implement digital passports (or identity cards), the government would have to issue many millions of digital passports. This requires a high security signature scheme that can be implemented efficiently. Using a suggestion as above ($k = l = 1000$ and $d = 3$), the signature itself is at most 4000 bits. Furthermore, each agent that verifies a signature needs only little storage itself (if one allows $O(k^2)$ bits storage for the agents, the schemes from [8] and [10] also easily deal with this application). This suggests that our scheme may very well eligible for such tasks.

References

1. W. Diffie, M. Hellman: *New Directions in Cryptography*, IEEE Transactions on Information Theory IT-22 (6): 644–654, 1976.

2. R. Rivest, A. Shamir, L. Adleman: *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, Communications of ACM, 21 (1978), pp. 120–126.

3. S. Goldwasser, S. Micali and R. Rivest: *A Digital Signature Scheme Secure Against Chosen Message Attacks*, SIAM Journal on Computing, 17(2): 281–308, 1988.

4. M. Bellare, S. Micali: *How to Sign Given any Trapdoor Function*, Proceedings of Crypto '88, Springer Verlag LNCS series, pp. 200–215.

5. O. Goldreich: *Two Remarks Concerning the GMR Signature Scheme*, Proceedings of Crypto '86, Springer Verlag LNCS series, pp. 104–110.

6. J. Bos, D. Chaum: *Provably Unforgeable Signatures*, Proceeding of Crypto '92, Springer Verlag LNCS series, pp. 1–14.

7. B. Pfitzmann: *Fail-Stop Signatures Without Trees*, Hildesheimer Informatik-Berichte 16/94, Universität Hildesheim, Juni 1994.

8. C. Dwork, M. Naor: *An Efficient Existentially Unforgeable Signature Scheme and its Applications*, Proceedings of Crypto'94, Springer Verlag LNCS series, pp. 218–238.

9. R. J. F. Cramer, I. B. Damgård: *Secure Signature Schemes based on Interactive Protocols*, BRICS Report Series RS–94–29, Aarhus University, Denmark, September '94 and Proceedings of Crypto '95, Springer Verlag LNCS series

10. R. J. F. Cramer: *Shared Randomness and the Size of Secure Signatures*, CWI Technical Report CS–R9530, April 1995.

11. R. J. F. Cramer, T. P. Pedersen: *Efficient and Provable Security Amplifications*, CWI Technical Report CS–R9529, April 1995.

12. R. C. Merkle: *A Certified Digital Signature*, Proceedings of Crypto '89, Springer Verlag LNCS series, pp. 234–246.

13. M. Naor, M. Yung: *Universal One-Way Hash Functions and Their Cryptographic Applications*, Proceedings of 21st STOC, 1989, pp. 33–43.

14. J. Rompel: *One-Way Functions are Necessary and Sufficient for Secure Signatures*, Proceedings of 22nd STOC, 1990, pp. 387–394.

15. T. ElGamal, *A Public-Key Cryptosystem and a Signature Scheme based on Discrete Logarithms*, IEEE Transactions on Information Theory, IT-31 (4): 469–472, 1985.

16. A. Fiat, A. Shamir: *How to Prove Yourself: Practical Solutions to Identification and Signature Problems*, Proceedings of Crypto '86, pp. 186–194

17. C. Schnorr: *Efficient Signature Generation by Smart Cards*, Journal of Cryptology, 4 (3): 161–174, 1991.

18. L. Guillou, J.J. Quisquater: *A Practical Zero-Knowledge Protocol fitted to Security Microprocessor Minimizing both Transmission and Memory*, Proceedings of Eurocrypt '88, Springer Verlag LNCS series, pp. 123–128.

19. T. Okamoto: *Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes*, Proceedings of Crypto '92, Springer Verlag LNCS series, pp. 31–53.

20. *Information Technology – Security Techniques – Digital Signature Scheme Giving Message Recovery*, ISO/IEC Standard 9796, first edition, International Standards Organization, Geneva.

21. National Institute of Technology and Standards: *Specifications for the Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication, US. Department of Commerce, 1993.

22. D. Chaum, T. P. Pedersen: *Wallet Databases with Observers*, Proceedings of Crypto '92, Springer Verlag LNCS series, pp. 89–105.

23. R. J. F. Cramer, T. P. Pedersen: *Improved Privacy in Wallets with Observers*, Proceedings of Eurocrypt '93, Springer Verlag LNCS series, pp. 329–343.

24. G. H. Hardy, E. M. Wright: *An Introduction to the Theory of Numbers*, fifth edition, 1979, Oxford Science Publications.

25. J. Benaloh, M. de Mare: *One-Way Accumulators: A Decentralized Alternative to Digital Signatures*, Proceedings of Eurocrypt '93, Springer Verlag LNCS series, pp. 274–285.