Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

On shared randomness and the size of secure signatures

R.J.F. Cramer

Computer Science/Department of Algorithmics and Architecture

CS-R9530 1995

Report CS-R9530
ISSN 0169-118X

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

On Shared Randomness and the Size of Secure Signatures

Ronald Cramer

CWI

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

Abstract

We present an efficient signature scheme that is not existentially forgeable under adaptively chosen message attacks [3]. The main feature of our scheme is that any practical number of signatures can be made while the size of the signatures remains relatively small, under the condition that all signers have access to a list of shared random strings.

More precisely, let integers l and d be fixed and let k be a security parameter. Given a list of l random $(k-1)$-bit strings shared by all signers, at least l^d signatures can be made by each signer in our scheme, where the size of a public key is k bits. The size of a signature does not exceed $(4d-3)k$ bits.

The first secure signature scheme where such trade-offs between shared randomness and the size of signatures has been realized was proposed by Dwork and Naor at Crypto '94 [1]. Their scheme is based on RSA, while their method for achieving efficiency relies on special properties of RSA that seem to go beyond the properties of general trapdoor permutations. Our contribution is to show that a secure signature scheme with similar efficiency can be based on a general cryptographic assumption that is potentially weaker than an RSA assumption, namely the existence of a family of claw-free trapdoor permutations [3], which can be constructed under the factoring assumption.

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

1. Introduction

A digital signature scheme is called secure if it is proven to be not existentially forgeable under adaptively chosen message attacks [3], relative to some standard cryptographic assumption. Briefly, this means that, under this cryptographic assumption, an attacker cannot forge a signature on a new message (i.e., a message that was not actually signed by the real signer), not even when he is allowed to use the signer as an oracle.

All practical digital signature schemes with well-established security in the sense of [3], have the rather unfortunate property that the size of a signature grows with the number of signatures made. Although this growth need not be more than $O(k \log i)$, where i is the number of signatures made, most secure signature schemes are presently not eligible for implementation in most cryptographic systems, especially those where smartcards are involved. The reason being that size of signatures in these schemes exceeds practical limits.

E-mail address: cramer@cwi.nl.

Dwork and Naor [1] were first to have constructed a more practical secure signature scheme based on RSA. Their scheme has the property that any practical number of signatures can be made, while the size of these signatures, though still being $O(k \log i)$, remains quite small. The price to be paid is a large list of random numbers shared by all signers.

This paper presents a secure digital signature scheme where the size of the signatures is also dramatically reduced compared to secure signature schemes preceding [1] at the cost of a list of random strings shared by all signers. Although signatures from [1] are shorter than ours, we have been able to construct a similarly efficient secure signature scheme based on the assumption that claw-free pairs of trapdoor permutations exist. This assumption is (potentially) weaker than an RSA-assumption, as it is known to hold under the factoring assumption. Briefly, the ideas behind our construction are as follows.

Using l pairs of claw-free trapdoor permutations, we authenticate l new nodes from using a given node. In this way, we build an authentication tree with branching degree l. Signing of a message is done in essentially the same way as in the GMR-scheme, using a separate claw-free trapdoor permutation pair. Note that the GMR-scheme employs one claw-free trapdoor permutation pair to construct an authenication with branching degree 2. We prove that this extension of the GMR-scheme is as secure as the GMR-scheme. Although this improves on the size of signatures, compared to GMR, as their size will now be proportional to $\log_l i$, instead of $\log_2 i$, the size of the public key is now large.

This matter is settled in our main result. There we present a secure signature scheme where the public-key of a signer only consists of one claw-free trapdoor permutation pair, while the size of signatures remains small. The idea is that, using a list of l random strings shared by all signers (as in [1]), and this permutation pair, a signer can now authenticate l pairs of claw-free trapdoor permutations. These $l+1$ pairs of permutations together with any of the shared strings define an instance of the extension outlined above. Using each of the l corresponding authentication trees with branching degree l up to depth $d-1$, the signer will be able to sign at least l^d messages. With respect to his public-key of k bits, signatures will be of size at most $(4d-3)k$ bits, where k is a security parameter. The size of the shared list is $l(k-1)$ bits.

In Sections 3 and 4, brief overviews of the schemes from [1] and [3] can be found. In Section 5.2, we give our extension of the GMR-scheme. The main result, our target digital signature scheme, is presented in Section 5.3. It is given in terms of an abstract family of claw-free pairs of trapdoor permutations, defined in Section 5.1. The performance figures, as given above, for the signature scheme in our main result, are achieved when it is implemented based on the difficulty of factoring integers, as demonstrated in Section 6.2. The family of permutations we present there is taylored to fit our purposes, by modifying the family of permutations based on factoring as given in [3]. A comparison with [3] and [1] is given in Section 7.

2. Notation

Let V be any set. Throughout this paper, "$v \leftarrow V$" means that v is selected uniformly at random from V, independent of anything else. If w is a member of V, "$v \leftarrow w$" means that the value w is assigned to v.

3. The Scheme of Dwork and Naor

The scheme from [1] works roughly as follows. Let public lists $X = \{x_0, \ldots, x_{l-1}\}$ and $Y = \{y_0, \ldots, y_{l-1}\}$ consist of l random k-bit numbers and of l primes, respectively. A signer selects a random RSA-modulus n and a random number $R \in \mathbb{Z}_n^*$, both of size k bits ($l \geq k$). The pair (n, R) is placed in the signer's public directory. The factorization of n is private input to the signer. With high probability, the elements in the list X will be relatively prime to n and the elements of Y will be relatively prime to $\phi(n)$.

Starting at the root R, the signer builds an authentication tree with branching degree l and depth d. All nodes are random elements of $\mathbb{Z}_n^*$. Let a be a node in the tree with a_j being its j-th child. The node a_j is authenticated by computing

$$(a \prod_{i=0}^{l-1} x_i^{b_i})^{\frac{1}{y_j}} \bmod n,$$

where $b_0 || \cdots || b_{l-1}$ is a binary representation of a_j (possibly padded with random bits). An authentication path for a leaf consists of a path to the root, together with all pairwise authentications as described above. Let an l-bit message $m = m_0 || \cdots || m_{l-1}$ be given, and let $\overline{a}$ be a leaf that has been used exactly $r-1$ times with $r - 1 < l$. The signature on m consists of an authentication path for $\overline{a}$ and the value

$$(\overline{a} \prod_{i=0}^{l-1} x_i^{m_i})^{\frac{1}{y_r}} \bmod n.$$

The leaves in the authentication tree are developed as needed. In this scheme, l^{d+1} messages can be signed, and the size of each signature is $(2d+1)k$ bits, whereas the size of the public key is $2k$ bits. The number of shared bits is lk for list X, plus the number of bits needed to encode the exponents in list Y. For details, see [1].

4. The GMR-Scheme

The digital signature scheme of Goldwasser, Micali and Rivest [3] is based on a family of claw-free pairs of trapdoor permutations. Such a family is known to exist under the factoring assumption. Informally, a pair of permutations $f = (f_0, f_1)$ with common domain D_f is called claw-free if it is infeasible to compute x and y in D_f such that $f_0(x) = f_1(y)$, given only the description of the permutations. A trapdoor for such a pair of claw-free permutations is a piece of information that enables efficient inversion of both f_0 and f_1. A claw-free pair of permutations f gives rise to a collision-free function as follows.

$$F \; : \; \{0,1\}^* \times D_f \; \rightarrow D_f$$

$$(a, x) \mapsto f_{a_0}(f_{a_1}(\cdots(f_{a_t}(x))\cdots)),$$

where $a = a_0||\cdots||a_t$ for some positive integer t. The function F applied to (a, x) will be denoted as $f_{[a]}(x)$. This function $f_{[\cdot]}(\cdot)$ is collision-free in the sense that it is infeasible to find $a, a' \in \{0,1\}^*$ and $x, y \in D_f$ such that $f_{[a]}(x) = f_{[a']}(y)$ and neither of the two strings a and a' is a prefix of the other. Observe that, given trapdoor information for f_0 and f_1, $f_{[a]}(z)$ can be computed efficiently for any $a \in \{0,1\}^*$ and any $z \in D_f$.

We will now briefly describe the signature scheme from [3]. On input of a security parameter k, a signer in the GMR-scheme generates two claw-free pairs $f = (f_0, f_1)$ and $g = (g_0, g_1)$, permuting sets D_f and D_g respectively [1], from the family, and also selects a random element R from D_f. The public key consists of the two claw-free pairs of permutations and R. The trapdoor information for the permutations is private input to the signer. The element R will be the root of a binary authentication tree of depth d (assuming that the expected number of signatures to be made does not exceed 2^d). The message space M is any subset of $\{0,1\}^*$ with the property that no string in M is a prefix of any other string in M.

We will only give an inductive decription of the scheme. Given a node x at depth at most $d-1$ in the authentication tree, two new nodes are generated as follows. First, the left child x_0 is chosen as a random element from D_f. The right child x_1 is then computed as

$$x_1 \leftarrow \left(f_{[x_0]}\right)^{-1}(x).$$

Each node at depth at most $d-1$ can only be used once in order to generate two new leaves as decribed above. An authentication path for a node consists of a list of all its ancestors and their children (note that this list contains the given node itself, by definition). In particular, an authentication path for a leaf consists of $2d$ members of D_f (excluding the root R, which is part of the public key anyway). New leaves can be constructed as needed.

Signing works as follows. Let a message $m = m_0||\cdots||m_t \in M$ be given. The signer selects a leaf $\overline{x}$ in the authentication tree, i.e., a node at depth d, and

1. selects a random element $z \in D_g$,
2. computes $\overline{z} \leftarrow \left(f_{[z]}\right)^{-1}(\overline{x})$
3. and $y \leftarrow \left(g_{[m]}\right)^{-1}(z)$,

Each leaf $\overline{x}$ is to be used only once to generate a signature. The signature on m consists of the values y, z, $\overline{z}$, and an authentication path for $\overline{x}$.

If an authentication tree of depth d is built, 2^d signatures can be made. The size of a signature is $(2d+3)k$ bits. The public key has size $3k$ bits.

[1] It is assumed that elements of these sets, as well as permutation pairs can be encoded in k bits

5. A New and Secure Signature Scheme with Small Signatures

In order to support the constructions to follow, we need a definition of a generator G for a family $\mathcal{F}$ of claw-free pairs of trapdoor permutations, that differs slightly from the definition given in [3]. In Section 6.2, we show that a family $\mathcal{F}$ of claw-free pairs of trapdoor permutations, satisfying our requirements, exists under the assumption that factoring integers is intractible.

Our main result is given in Section 5.3, and uses the results presented in Section 5.2

5.1 The Generator G

Our definition of a family of claw-free pairs of trapdoor permutations below is obtained by taking that from [3], and adding requirement 2. We will assume that we are given a probabilistic polynomial time generator G for a family $\mathcal{F}$ of claw-free pairs of trapdoor permutations, with the following properties.

1. On input of 1^k, G outputs a pair of permutations $f = (f_0, f_1)$ of a set D_f, and the corresponding trapdoor information s_f that enables efficient inversion of both f_0 and f_1. It is assumed that we can sample elements of D_f with uniform distribution over D_f.

2. An efficient embedding [2] $\kappa_f : \{0,1\}^{\overline{k}} \longrightarrow D_f$ is available, where $\overline{k}$ only depends on k. Furthermore, $\frac{2^{\overline{k}}}{\#D_f} > \frac{1}{k^c}$ for some constant c and sufficiently large k.

3. (*Claw-Freeness*)
   No probabilistic polynomial time algorithm can, on input of a pair (f_0, f_1) only (i.e., the trapdoor information s_f is not part of the input), as generated by $G(1^k)$, output $x, y \in D_f$ such that $f_0(x) = f_1(y)$, except with negligible probability of success (in k).

Note that the second requirement guarantees that for k sufficiently large, the probability that a random element of D_f is in the image of κ_f, is non-negligible for each f as output by $G(1^k)$.

See also Section 6.2, where a generator G based on the difficulty of factoring integers, is exhibited. There, we have in fact $\overline{k} = k - 1$ and $2^{k-1} < \#D_f < 2^k$ for each f as output by $G(1^k)$.

5.2 An Extension of the GMR-Scheme

In this section we will describe an extension of the GMR-scheme. At the expense of a large public key $(l+2)k$ bits rather than $2k$ bits, we will demonstrate how to obtain a secure signature scheme where the size of the i-th signature is proportional to $\log_l i$, instead of

[2] It is also assumed that, given an element y in the image of κ_f, one can efficiently compute x such that $\kappa_f(x) = y$. Furthermore, for our purposes, it is sufficient if κ_f is defined on all but a negligible fraction of $\{0,1\}^{\overline{k}}$.

$\log_2 i$ as in the GMR-scheme. This means that if l is taken large, signatures become much shorter than in the GMR-scheme.

The signature scheme Σ to follow can be easily obtained from the construction of [3] as outlined in Section 4. The difference between Σ and the GMR-scheme lies in the way the authentication tree is built. Let integers l and d be given.

We will use l claw-free trapdoor permutation pairs

$$f^{(0)} = (f_0^{(0)}, f_1^{(0)}), \ldots, f^{(l-1)} = (f_0^{(l-1)}, f_1^{(l-1)})$$

to generate to create l new leaves from a given one, rather than using one such pair to create 2 new leaves from a given one as in the GMR-scheme.

In this way, we will build a full authentication tree with branching degree l and depth d. As before, a random string $R \in \{0,1\}^{\overline{k}}$ will be the root, and messages are signed using a claw-free trapdoor permutation pair $g = (g_0, g_1)$ (see Section 4). Details about signing can be found in the following. Furthermore, all nodes in the authentication tree can be used to make signatures (in contrast to the GMR-scheme where only the leaves can be used for this purpose). The public key of the signer in signature scheme Σ thus consists of

$$(R, g, f^{(0)}, \ldots, f^{(l-1)}).$$

The trapdoor information to these permutation pairs is private input to the signer. The permutation pairs and their corresponding trapdoor information are obtained by running $G(1^k)$ $l+1$ times.

The authentication tree will be built up as follows. Suppose that we have an authenticated leaf x at depth at most $d-1$. From this leaf, we will generate l new ones, by generating l random $\overline{k}$-bitstrings $x_0, \ldots, x_{l-1}$. These new leaves x_j are authenticated by computing

$$y_j \leftarrow \left(f_{[x_j]}^{(j)}\right)^{-1} (\kappa_{f^{(j)}}(x)),$$

for $j = 0, \ldots, l-1$. This process of building up an authentication tree is started at the root, and new nodes can be created as needed. An authentication path for a node x is a list consisting of x, all its ancestors, and all authentication values between adjacent nodes from the list. For the sake of concreteness, it is assumed that the depth of the tree will be at most d. Note that such an authentication tree of branching degree l consists of at least l^d nodes.

A difference with the GMR-scheme is that in the GMR-scheme, where only one permutation pair f is used to authenticate new nodes, the authentication values of new nodes (i.e., the values x_1 in our description of the GMR-scheme. See Section 4) can be viewed as nodes in the authentication tree themselves, and can as such be used to serve as the parent of new nodes. In our scheme where many such pairs (i.e., the pairs $(f_0^{(i)}, f_1^{(i)})$ for $i = 0, \ldots, l-1$, where l is typically large) are used for authentication of new nodes, this kind of re-using seems impossible for technical reasons in the proof of security.

The signing process works as follows. For simplicity, let the messagespace be here be $\{0,1\}^t$ for some positive integer t. Given a message m to be signed and a node $\overline{x} \neq R$ at depth at most d, the signer computes

$$y \leftarrow \left(g_{[m]}\right)^{-1}(\kappa_g(\overline{x})),$$

where $m = m_0||\cdots||m_{t-1}$ is the message to be signed. Each such node $\overline{x}$ in the authentication tree, is to be used only once to generate a signature.

Remark 1 *An important remark about the order in which such nodes $\overline{x}$ are used for making signatures as described above, is necessary here. Although the signer can create new nodes in the authentication tree as he needs them and although it does not matter if he develops the tree in depth-first order, breadth-first order or any other way he chooses, it is imperative, from a security point of view, that it is the first time that $\overline{x}$ is made public when $\overline{x}$ is used for making a signature as above (a similar requirement applies to the scheme from [3]).*

More technically, if $(y, auth(\overline{x}))$ is a signature on message m that the signer presents to a receiver, it must be the case that, for any previous signature $(y', auth(x'))$ on some message m', he presented, $\overline{x}$ is not a node in $auth(x')$. The reason can be found in the proof of security (see case $r = -1$). In our description of Σ to follow this requirement is satisfied.

From a storage point of view, it is most convenient to create and use new nodes in depth-first order, as in that case at most d nodes have to be stored at any time.

Remark 2 *Besides the pair g, at most d out of the remaining l permutation pairs that are contained in the public key, are needed for the generation and verification of a signature in Σ.*

The message space for the signature scheme Σ is assumed to be the set $M(k) \equiv \{0,1\}^{P(k)}$, where $P(k)$ is any (strictly positive) polynomial in k. Furthermore, let l and d be fixed positive integers.

The Signature Scheme Σ :

Initialization: On input of a security parameter k, the signer generates $l+1$ independent pairs of claw-free trapdoor permutations

$$g = (g_0, g_1), f^{(0)} = (f_0^{(0)}, f_1^{(0)}), \ldots, f^{(l-1)} = (f_0^{(l-1)}, f_1^{(l-1)}),$$

with known trapdoor information, by running $G(1^k)$ $l+1$ times. The signer selects a random $\overline{k}$-bit string R and defines his public key as

$$pk \leftarrow (R, g, f^{(0)}, \ldots, f^{(l-1)}).$$

The trapdoor information of all these permutation pairs involved is private input to the signer.

Signing: The following description is inductive. Before the first signature request, the signer starts with the authentication tree consisting of the root R only. Let x be any node in the authentication tree at depth at most $d-1$, with the following property: x does not have a child authenticated with respect to $f^{(j)}$ for some $j = 0, \ldots l-1$. Then the signer selects a random string $x_j \in \{0,1\}^{\overline{k}}$. This new node x_j is authenticated by computing

$$y_j \leftarrow \left(f^{(j)}_{[x_j]}\right)^{-1} (\kappa_{f^{(j)}}(x)).$$

Given a message $m \in M(k)$, the signer computes

$$y \leftarrow \left(g_{[m]}\right)^{-1} (\kappa_g(x_j))$$

This signature in Σ uses at most d', with $d' \leq d$, out of the $f^{(i)}$ $(i = 0, \ldots, l-1)$, say, those corresponding to $i_1, \ldots, i_{d'}$. Note that, besides R and the pair g, only these d' permutation pairs are needed to verify $\sigma_{pk}(m)$.

The signature $\sigma_{pk}(m)$ then consists of

$$\sigma_{pk}(m) \leftarrow (y, \text{auth}(x_j)),$$

where $\text{auth}(x)$ is an authentication path for x .

Verification: The verifier checks whether $g_{[m]}(y) = \kappa_g(x_j)$ and whether $\text{auth}(x_j)$ is a valid authentication path for x_j, given pk. If all verifications are satisfied, $\sigma_{pk}(m)$ is accepted as a valid signature on the message m with respect to public key pk.

Remark 3 *In the following, it is assumed that $\overline{k} \leq k$ and that the elements of domains of claw-free permutation pairs, as well as the pairs themeselves can be encoded in k bits.*

Remark 4 *Note that by our assumptions on the embeddings κ, we may assume that each of our $l+1$ embeddings κ is defined on all but a negligible fraction of $\{0,1\}^{\overline{k}}$. Thus, the probability that a random $x \in \{0,1\}^{\overline{k}}$ has the property that one of the embeddings κ is undefined at x, can be neglected. Therefore, in the analysis to follow, it will simply be assumed that all embeddings have domain equal to $\{0,1\}^{\overline{k}}$.*

Theorem 1 *Let l and d be fixed positive integers. Suppose that a family $\mathcal{F}$ of claw-free pairs of trapdoor permutations exists with generator G as in Section 5.1. Then the signature scheme Σ, described above, is not existentially forgeable under adaptively chosen message attacks. Let k be the security parameter. With respect to a public key of size $(l+2)k$ bits, at least l^d signatures can be made. The size of a signature is at most $(2d+1)k$ bits.*

Proof: Reasoning by contradiction, we show that a successful attacker of the signature scheme Σ can be compiled into a successful algorithm that breaks the claw-freeness of the family $\mathcal{F}$ of claw-free pairs of trapdoor permutations. Our algorithm for computing

claws, running the attacker as a subroutine, will have almost the same success probability as the attacker of Σ itself.

To this end, the attacker gets as input a signer in the signature scheme Σ, and is allowed to make at most N calls, i.e., he is entitled to request signatures on at most N messages chosen in an adaptive fashion, where $N = \sum_{i=1}^{d} l^i$. After this process has been completed, we may assume, the attacker outputs a forgery on a new message $\tilde{m}$ whose signature has not been requested by the attacker.

Without loss of generality, we may assume that the attacker outputs the forgery after exactly N calls to the signer: if he can compute a forgery after $N' < N$ calls, we simply redefine the attacker so that he will make an additional $N - N'$ calls on a message that is not equal to $\tilde{m}$.

The compilation works as follows. Suppose we are given a pair $h = (h_0, h_1)$ from the family $\mathcal{F}$ as generated by the generator G on input 1^k, but not its trapdoor information s_h. We will use the attacker to compute a claw for this pair h. We start by running $G(1^k)$ l times in order to obtain a list of l claw-free pairs of trapdoor permutations with known trapdoor information.

Second, we select a random $r \in \{-1, 0, \ldots, l-1\}$. If $r > -1$, we will add the pair h to the list such that it becomes the $r+2$-nd entry. The resulting list of $l+1$ pairs consists of $g = (g_0, g_1)$, $f^{(0)} = (f_0^{(0)}, f_1^{(0)}), \ldots, f^{(l-1)} = (f_0^{(l-1)}, f_1^{(l-1)})$, where $h = f^{(r)}$. If $r = -1$, we put h at the head of the list such that $h = g$.

Next we demonstrate that in all cases r with $-1 \leq r \leq l-1$, we can set up a signer for the signature scheme Σ, whose permutation pairs are those in the list of $l+1$ claw-free pairs of trapdoor permutations as defined above, such that one cannot distinguish between a real signer and this "simulated" signer (who doesn't know the trapdoor information for exactly one of the $l+1$ pairs). In particular, it follows from this (perfect) simulation that all cases are indistinguishable from one another.

Running the attacker on this simulated signer, we may expect the attacker to produce a forgery with essentially the same success probability as in real life, by the perfectness of the simulation. Our argument is completed by the observation that from a forgery and the signatures output as a result of the attacker's calls to the simulator, we can derive a claw for at least one of the $l+1$ permutation pairs. By the perfectness of the simulation, this forgery leads to a claw exactly for the pair h, for which we were not given the trapdoor information, with probability $\frac{1}{l+1}$.

As argued above, it is now sufficient to describe our simulation, and demonstrate the perfectness of the simulation and prove that a forgery leads to a claw for one of the $l+1$ permutation pairs.

$r > -1$:
: We have to build a full authentication tree of branching degree l and depth d (clearly consisting of $N+1$ nodes) while the trapdoor information to $f^{(r)}$ is not given for

exactly one r with $0 \leq r \leq l-1$. This tree is now built from the bottom up. Observe that it works for trees of depth 1 as follows.

Choose $x_0, \ldots, x_{l-1} \leftarrow \{0,1\}^{\overline{k}}$. Then, select $y_r \leftarrow D_{f^{(r)}}$ and compute $f^{(r)}_{[x_r]}(y_r)$. This value is distributed uniformly over $D_{f^{(r)}}$ and is independent of $x_0, \ldots, x_{l-1}$. If this value is in the image of $\kappa_{f^{(r)}}$, we compute $x \in \{0,1\}^{\overline{k}}$ such that $\kappa_{f^{(r)}}(x) = f^{(r)}_{[x_r]}(y_r)$ and we halt. Otherwise, we keep selecting $y_r \leftarrow D_{f^{(r)}}$ until this is the case. Note that, in any case, x is distributed uniformly over $\{0,1\}^{\overline{k}}$ and is independent of $x_0, \ldots, x_{l-1}$. By our assumptions on the embeddings (see Section 5.1), the process just described has non-negligible success probability of being completed in polynomial time. After this, for $i = 0, \ldots, l-1$ and $i \neq r$, y_i is computed as $y_i \leftarrow \left(f^{(i)}_{[x_i]}\right)^{-1}(\kappa_{f^{(i)}}(x))$. Note that all values y_i, for $i = 0, \ldots, l-1$, follow deterministically from $x, x_0, \ldots, x_{l-1}$. The value x is to be viewed as the root R, and $x_0, \ldots, x_{l-1}$ as its children.

Supose now, as an induction hypothesis, that we can build such authentication trees with depth $n-1$. Now, given l such trees, we can easily build an authentication tree of depth n by putting the roots of these l trees of depth $n-1$ in the roles of $x_0, \ldots, x_{l-1}$ as above and compute the root R in the same way as x was computed above.

All nodes in the authentication trees thus constructed have uniform distribution over $\{0,1\}^{\overline{k}}$ and are independent of each other and anything else. The corresponding authentication values follow deterministically.

After a full authentication tree of depth d and branching l has thus been constructed and the attacker is given the public key, the simulator can start answering the calls by the attacker, as the trapdoor information to the pair g is given in this case where $r > -1$. As the real signer, the simulator will only reveal a node $\overline{x}$ that the attacker has not seen before if at the same time a signature $y \leftarrow \left(g_{[m]}\right)^{-1}(\kappa_g(\overline{x}))$ for some request m is revealed.

$r = -1$:

In this case, we are given the trapdoor information for the pairs $f^{(i)}$, for $i = 0, \ldots, l-1$, but not the trapdoor information for the pair g. This time, we will select the root R for the authentication tree as in the description of Σ and present the resulting public key to the attacker.

Next, we construct the authentication tree top down interleaved with the requests from the attacker, as follows. Given a request for a signature on a message m, we simply compute $g_{[m]}(y)$ for $y \leftarrow D_g$ and with non-negligible probability, $g_{[m]}(y)$ is in the image of κ_g and we compute $x \in \{0,1\}^{\overline{k}}$ such that $\kappa_g(x) = g_{[m]}(y)$. Finally, we authenticate x as in the description of Σ. In this way, the simulator is able to answer all (at most N) calls by the attacker. Again, as in the case $r > -1$, the nodes in the authentication tree have uniform distribution over $\{0,1\}^{\overline{k}}$ and are independent of each other and anything else. Also, the corresponding authentication values follow deterministically. Note that this simulation has the property that when a non-root

node is made available to the attacker, we must give a signature with respect to this node immediately, or not be able to do so for future requests (as we are not given the trapdoor information for the pair g, we must know the message before the node is created).

From the above analysis we get that the simulation is perfect and runs in probabilistic polynomial time. Next, we may assume that the attacker outputs a forgery with some probability ϵ.

Let the forgery, on a new message $\tilde{m}$, be a pair $(\tilde{y}, \text{auth}(\tilde{x}))$, where $g_{[\tilde{m}]}(\tilde{y}) = \kappa_g(\tilde{x})$ and $\text{auth}(\tilde{x})$ is a valid authentication path for $\tilde{x}$, i.e., a list of elements from $\{0,1\}^{\overline{k}}$, starting at the root R and ending with $\tilde{x}$, together with pairwise authentications of these elements (by definition of a valid signature, $\tilde{x}$ will be different from the root R).

There are two possibilities. First, with probability ϵ_1, $\tilde{x}$ is equal to one of the nodes x of the authentication tree. Then, since $\tilde{m}$ was not signed by the simulator ($\tilde{m}$ is "new"), we have immediately a claw, as we have assumed that the simulator has made signatures with respect to all its N non-root nodes. Therefore, there are $m \in M(k)$ and $y \in D_g$ available as a result of the calls to the simulator, such that $g_{[m]}(y) = g_{[\tilde{m}]}(\tilde{y}) = \kappa_g(x) = \kappa_g(\tilde{x})$ and $m \neq \tilde{m}$. This clearly results in a claw for the pair g.

On the other hand, with probability ϵ_2 ($\epsilon_1 + \epsilon_2 = \epsilon$), $\tilde{x}$ is new. But, clearly, $\text{auth}(\tilde{x})$ must contain at least one node from the authentication tree output by the simulation (since it must trace back to the root R). Let x be the youngest such node in $\text{auth}(\tilde{x})$ (i.e., the one closest to $\tilde{x}$ in $\text{auth}(\tilde{x})$). Then x must have a child x' in $\text{auth}(\tilde{x})$, and moreover this child is not in the authentication tree output by the simulator. Let x' be authenticated in $\text{auth}(\tilde{x})$ with respect to the pair $f^{(j)}$ for some j with $0 \leq j \leq l-1$ and with authentication value y', and let x_j be the child of x in the authentication tree output by the simulator, authenticated also with respect to the pair $f^{(j)}$ and with authentication value y_j (again, this node and the authentication value are available, since we have assumed that the simulator uses up a full authentication tree). This results in a claw for the pair $f^{(j)}$, since we have $f^{(j)}_{[x_j]}(y_j) = f^{(j)}_{[x']}(y') = \kappa_{f^{(j)}}(x)$ and $x_j \neq x'$.

The proof is completed by observing that, from the perfectness of the simulation, with probability at least $\frac{\epsilon}{l+1}$, the claw we have obtained will be with respect to the pair h, for which we were not given the trapdoor information.

□

Remark 5 *At any time, a signer can release $\alpha \in M(k)$ and $\beta \in D_g$ such that $g_{[\alpha]}(\beta) = \kappa_g(R)$, provided that α is chosen by the signer, independently of R. It is easily verified that the proof of security in this case still holds: in the case where the trapdoor to g is not given in the proof, R will be computed as above for random $\beta \in D_g$, instead of choosing R as a random $\overline{k}$-bit string. This property of Σ is exploited in our main result in Section 5.3.*

5.3 Main Result

Our final goal is to reduce the size of the public key in signature scheme Σ, while (almost) preserving the size of signatures. To this end, we modify Σ and obtain our target scheme Σ^*, where the public key of a signer only consists of one claw-free trapdoor permutation pair (instead of $l+1$). If we assume again that a permutation pair can be encoded with k bits, this saves lk bits. This extension Σ^* is available at the cost of (almost) doubling the size of signatures (compared to the length of signatures in Σ) and a list with l random $\overline{k}$-bitstrings shared by all signers. As a result, the overall storage needed in the system is cut down by, roughly, a factor l.

The idea is the following. Given this list of l shared random strings, and a claw-free permutation pair g with known trapdoor information, a signer will create an instance of Σ, by generating l independent claw-free permutation pairs $f^{(i)}$, $i = 0, \ldots, l-1$, with known trapdoor information, and having an element S of the shared list as the root of an authentication tree of branching degree l and depth $d-1$. As shown in Section 5.2, this allows him to make $l + \cdots + l^{d-1}$ signatures in Σ, with respect to public key $(S, g, f^{(0)}, \ldots, f^{(l-1)})$.

But, there are l elements in the list available, each of which can serve as the root for a new authentication tree. So, if one tree is used up, the signer switches to the next by selecting a new element S' from the shared list and doing the same thing over again. However, he keeps the $l+1$ permutation pairs that he selected before. Thus the signatures in Σ in this new case after switching to S' will be with respect to public key $(S', g, f^{(0)}, \ldots, f^{(l-1)})$.

The public key in the target scheme Σ^* consists only of the pair g. To describe what the resulting signature Σ^* with respect to this public key entails, we only have to show how the permutation pairs $f^{(i)}$ are authenticated. Suppose the signer has computed a signature in Σ with respect to public key $(S, g, f^{(0)}, \ldots, f^{(l-1)})$ as above. From Remark 2, at most $d-1$ out of the l $f^{(i)}$'s have to be authenticated. This is done using the result mentioned in Remark 5. More precisely, if $f^{(j)}$ needs authentication for some j, this is simply achieved by computing

$$\beta_j \leftarrow \left(g_{[\alpha_j]}\right)^{-1}(\kappa_g(S_j)),$$

where α_j is a suitable encoding of the pair $f^{(j)}$ and S_j is the j-th element in the shared list. For these at most $d-1$ pairs, the authentications (including the descriptions of these pairs) cost at most $2(d-1)k = (2d-2)k$ bits, whereas the corresponding signature in Σ costs at most $(2(d-1)+1)k = (2d-1)k$ bits.

In this particular example, the signature in Σ^* with respect to public key g now consists of a signature in Σ with respect to public key $(S, g, f^0, \ldots, f^{l-1})$ and the authentications for the (at most) $d-1$ $f^{(i)}$'s needed for the verification of this signature.

So, with only one claw-free permutation pair g with known trapdoor information and the shared list at hand, the signer in Σ^* can make $l^2 + \cdots + l^d$ signatures, whose size does not exceed $(4d-3)k$ bits.

Note that a sloution where the GMR-scheme is implemented, with the exception that all signers use the same l random strings as the roots for l authentication trees, where they

switch to the next tree if the signatures become to large, gives much larger signatures than in Σ^*.

More precisely, the signature scheme Σ^* is defined as follows. Let l and d be fixed positive integers. Suppose we are given a family $\mathcal{F}$ of claw-free pairs of trapdoor permutations, with generator G as in Section 5.1. Let k be the security parameter, and let the list X, shared by all signers, consist of l random $\overline{k}$-bit strings $\{S_0, \ldots, S_{l-1}\}$. It is assumed that this list is generated in a way trusted by all signers. The message space for the signature scheme Σ^* is assumed to be the set $M(k) = \{0,1\}^{P(k)}$, where $P(k)$ is any (strictly positive) polynomial in k.

The Signature Scheme Σ^* :

Initialization: The signer generates $l+1$ independent pairs of claw-free trapdoor permutations

$$g = (g_0, g_1), f^{(0)} = (f_0^{(0)}, f_1^{(0)}), \ldots, f^{(l-1)} = (f_0^{(l-1)}, f_1^{(l-1)}),$$

with known trapdoor information, by running $G(1^k)$ (see Section 5.1) $l+1$ times. The signer puts

$$pk_i \leftarrow (S_i, g, f^{(0)}, \ldots, f^{(l-1)}),$$

and computes

$$\beta_i \leftarrow \left(g_{[\alpha_i]}\right)^{-1} (\kappa_g(S_i)),$$

for $i = 0, \ldots, l-1$, where α_i is a suitable encoding of the pair $f^{(i)}$ in k bits. His public key in our signature scheme Σ^* is now

$$pk^* \leftarrow g.$$

The trapdoor information to all these claw-free permutation pairs is private input to the signer.

Signing: Given a message $m \in M(k)$, the signer selects i with $0 \leq i \leq l-1$, subject to the condition that he has selected i at most $l + \cdots + l^{d-1} - 1$ times before. Next, he computes a signature $\sigma_{pk_i}(m)$ in Σ (see Section 5.2) with respect to public key pk_i. Here, the branching degree is l and the depth of the authentication tree is $d-1$. This signature in Σ uses at most d', with $d' \leq d-1$, out of the $f^{(i)}$ $(i = 0, \ldots, l-1)$, say, those corresponding to $i_1, \ldots, i_{d'}$. Note that, besides g, the only permutation pairs needed to verify $\sigma_{pk_i}(m)$ are exactly these. See also Remark 2. The signature $\sigma_{pk^*}(m)$ in Σ^* then consists of

$$\sigma_{pk^*}(m) \leftarrow (\sigma_{pk_i}(m), \alpha_{i_1}, \beta_{i_1}, \ldots, \alpha_{i_{d'}}, \beta_{i_{d'}}).$$

Verification: First, the verifier retrieves S_i, $S_{i_1}, \ldots, S_{i_{d'}}$ from the public list X. From the α_{i_j}, he computes the descriptions of $f^{(i_j)}$ for $j = 1, \ldots, d'$.

Finally, he checks whether $\sigma_{pk_i}(m)$ is a valid signature on m in Σ, given S_i, g and $f^{(i_j)}$ for $j = 1, \ldots, d'$ and whether $g_{[\alpha_{i_j}]}(\beta_{i_j}) = \kappa_g(S_{i_j})$, for $j = 1, \ldots, d'$. If all verifications are satisfied, σ_{pk^*} constitutes a valid signature on message m in Σ^*, with respect to public key pk^*.

We have the following theorem. The proof of the theorem is similar to that of Theorem 1.

Theorem 2 *Let l and d be fixed positive integers. Suppose that a family $\mathcal{F}$ of claw-free pairs of trapdoor permutations exists with generator G as in Section 5.1. Then the signature scheme Σ^*, described above, is not existentially forgeable under adaptively chosen message attacks. Let k be the security parameter, and let a list consisting of l random $\overline{k}$-bit strings be shared by all signers. With respect to a public key of size k bits, at least l^d signatures can be made. The size of a signature is at most $(4d-3)k$ bits.*

Proof: See the proof of Theorem 1. In case $r > -1$, the list X is generated by applying the simulation for case $r > -1$ exactly l times in order to obtain l independent authentication trees with branching degree l and depth $d-1$. If on the other hand, $r = -1$, S_i is computed as $S_i \leftarrow g_{[\alpha_i]}(\beta_i)$, where β_i is chosen at random from D_g such that S_i is in the image of κ_g.

Our analysis for finding claws has to be augmented with the case that the forgery contains a permutation pair not authenticated by the signer. This case leads to a claw for g. The rest of the proof is similar to the proof of Theorem 1. □

If we compare this scheme to that of the previous section, we observe that the size of the public directory has been dramatically reduced by roughly a factor of l at the cost of (almost) doubling the size of signatures.

Remark 6 *The parameters l and d have only been chosen to be fixed for reasons of clarity. These parameters can also be chosen as a polynomial function of k. Moreover, they can even be changed after an instance of the signature scheme is generated. As an example, the signer need not halt if he has reached depth $d-1$. Also the shared list may be expanded. We leave it to the reader to find out about the possible variations on our description. This remark also holds for the signature scheme Σ from Section 5.2.*

6. An Implementation based on the Difficulty of Factoring Integers

In Section 5.1, we defined a generator G for a family of claw-free trapdoor permutations suitable for the construction of our main result, the signature scheme Σ^*. This definition was obtained by adding a requirement to the definition of such a generator as given in [3].

One can easily observe that the implementation of the family of claw-free trapdoor permutations based based on the difficulty of factoring integers from [3], does not satisfy our requirements.

In this section, we will define a family of claw-free trapdoor permutations based on factoring, that fits with the requirements of Section 5.1. We start by giving an overview of the family of claw-free trapdoor permutations based on factoring as defined in [3]. This family of permutations is then modified so as to suit our purposes.

6.1 Claw-Free Trapdoor Permutations based on Factoring as in GMR

Let n be an integer that is the product of two primes p and q, such that $p \equiv 3 \pmod 8$ and $q \equiv 7 \pmod 8$. Then, -1 is a non-quadratic residu modulo n, $J_n(-1) = 1$ and $J_n(2) = -1$, where J_n denotes the Jacobi-symbol. In the following, we will refer to these integers n as Blum-integers.

For any Blum integer n define

$$D_n = \{0 < x < \frac{n}{2} \mid J_n(x) = 1\}.$$

From n, a pair of claw-free permutations f_0 and f_1 of D_n is constructed as follows.

$$f_0 : D_n \longrightarrow D_n$$

$$x \mapsto \begin{cases} x^2 \pmod n : 0 < x^2 \pmod n < \frac{n}{2} \\ -x^2 \pmod n : \frac{n}{2} < x^2 \pmod n < n, \end{cases}$$

$$f_1 : D_n \longrightarrow D_n$$

$$x \mapsto \begin{cases} 4x^2 \pmod n : 0 < 4x^2 \pmod n < \frac{n}{2} \\ -4x^2 \pmod n : \frac{n}{2} < 4x^2 \pmod n < n. \end{cases}$$

Note that, given the factorization of n, f_0 and f_1 can be efficiently inverted. Furthermore, in [3] it is shown that given $x, y \in D_n$ such that $f_0(x) = f_1(y)$ (i.e., a "claw"), the factorization of n can be efficiently computed. For details, see [3].

6.2 Modifying the GMR Claw-Free Permutations based on Factoring

Let a Blum integer n be given as before. We will now demonstrate how we can construct a pair of claw-free permutations $\overline{f_0}$ and $\overline{f_1}$ of $\mathbb{Z}_n^*$, from a pair of claw-free permutations (f_0, f_1) of D_n.

The basic idea is to partition $\mathbb{Z}_n^*$ into four pairwise disjoint subsets of the same size, one of which is D_n, and define bijections between these sets and D_n.

For $b = 0, 1$, $\overline{f_b}$ is defined as follows. Given $x \in \mathbb{Z}_n^*$, map x into D_n with the relevant bijection (depending on which of the subsets x is in) apply f_b, and map the result back into the subset that x is in. It can be easily understood that this yields claw-free permutations $\overline{f_0}$ and $\overline{f_1}$ of $\mathbb{Z}_n^*$, if f_0 and f_1 are claw-free permutations of D_n.

More concretely, define the following.

$$
\begin{aligned}
D_1 &= D_n \\
D_2 &= \{0 < x < \tfrac{n}{2} \mid J_n(x) = -1\} \\
D_3 &= \{\tfrac{n}{2} < x < n \mid J_n(x) = 1\} \\
D_4 &= \{\tfrac{n}{2} < x < n \mid J_n(x) = -1\},
\end{aligned}
$$

$$
\sigma : \mathbb{Z}_n^* \longrightarrow \{1,2,3,4\}
$$

$$
x \mapsto \begin{cases} 1 & : \quad x \in D_1 \\ 2 & : \quad x \in D_2 \\ 3 & : \quad x \in D_3 \\ 4 & : \quad x \in D_4, \end{cases}
$$

$$
\tau_1 : D_1 \longrightarrow D_1
$$

$$
x \mapsto x,
$$

$$
\tau_2 : D_2 \longrightarrow D_1
$$

$$
x \mapsto \begin{cases} 2x \pmod n : 0 < 2x \pmod n < \frac{n}{2} \\ -2x \pmod n : \frac{n}{2} < 2x \pmod n < n, \end{cases}
$$

$$
\tau_3 : D_3 \longrightarrow D_1
$$

$$
x \mapsto -x \pmod n,
$$

$$
\tau_4 : D_4 \longrightarrow D_1
$$

$$
x \mapsto \begin{cases} 2x \pmod n : 0 < 2x \pmod n < \frac{n}{2} \\ -2x \pmod n : \frac{n}{2} < 2x \pmod n < n. \end{cases}
$$

Observe that D_1, D_2, D_3 and D_4 partition $\mathbb{Z}_n^*$ into 4 disjoint sets of the same size and that τ_1, τ_2, τ_3 and τ_4 are bijective. It also follows that the indicator function σ is well-defined for all $x \in \mathbb{Z}_n^*$.

$$
\overline{f_0} : \mathbb{Z}_n^* \longrightarrow \mathbb{Z}_n^*
$$

$$
x \mapsto \tau_{\sigma(x)}^{-1} \circ f_0 \circ \tau_{\sigma(x)}(x),
$$

$$
\overline{f_1} : \mathbb{Z}_n^* \longrightarrow \mathbb{Z}_n^*
$$

$$
x \mapsto \tau_{\sigma(x)}^{-1} \circ f_1 \circ \tau_{\sigma(x)}(x),
$$

Lemma 1 *$\overline{f_0}$ and $\overline{f_1}$ constitute a pair of claw-free permutations of $\mathbb{Z}_n^*$, if f_0 and f_1 constitute a pair of claw-free permutations of D_n.*

Proof: Observe that $\overline{f_0}$ and $\overline{f_1}$, when restricted to D_i, are in fact permutations of D_i for $i = 1, 2, 3, 4$. Now suppose that we can compute $x, y \in \mathbb{Z}_n^*$ such that

$$\overline{f_0}(x) = \overline{f_1}(y),$$

given only n, but not its factorization. It follows that

$$\tau_{\sigma(x)}^{-1} \circ f_0 \circ \tau_{\sigma(x)}(x) = \tau_{\sigma(y)}^{-1} \circ f_1 \circ \tau_{\sigma(y)}(y)$$

But then, we must have

$$\sigma(x) = \sigma(y),$$

by the first remark and the fact that all D_i are disjoint. Therefore, we have

$$f_0(\tau_i(x)) = f_1(\tau_i(y)),$$

with $\tau_i(x), \tau_i(y) \in D_n$ for some $i \in \{1, 2, 3, 4\}$, and we have created a claw for f_0 and f_1. □

Note that $\overline{f_0}$ and $\overline{f_1}$ can efficiently be inverted when given the factorization of n.

6.3 Definition of Generator G

We now define our family $\mathcal{F}$ of claw-free pairs of trapdoor permutations based on the difficulty of factoring integers, by defining a suitable generator G (See Section 5.1).

Let $N(k)$ denote the set of all Blum-integers n such that $2^{k-1} < n < 2^k$. Let $n \in N(k)$, and let $f = (f_0, f_1)$ be the pair of claw-free permutations of $\mathbb{Z}_n^*$ as defined in Section 6.2. Define $E(k)$ to be the set of integers x such that $0 \leq x < 2^{k-1}$, and put $\overline{k} = k - 1$. Suppose that the prime factors of n have size approximately equal to $\sqrt{n}$. Then:

1. All but a negligle fraction of E_k is contained in $D_f = \mathbb{Z}_n^*$.
2. $\frac{\#E_k}{\#D_f} \geq \frac{1}{2} - \epsilon$, where ϵ is negligible in k.

Definition 1 *(Generator G)*
On input of 1^k, G outputs a random $n \in N(k)$ together with its prime factors p and q, subject to the condition that these prime factors have size approximately $\sqrt{n}$. The corresponding claw-free permutation pair $f = (f_0, f_1)$ is defined as in Theorem 2. The trapdoor information s_f consist of the factorization of n, i.e., p and q. The embedding $\kappa_f : \{0,1\}^{k-1} \longrightarrow D_f = \mathbb{Z}_n^*$ is defined by identifying $E(k)$ with $\{0,1\}^{k-1}$ and mapping $x \in E(k)$ to $x \pmod n$. Claw-freeness is satisfied if factoring integers is intractible. The family of claw-free pairs of trapdoor permutations, generated by G, is denoted $\mathcal{F}_{fact}$.

Remark 7 *The embedding k_f is defined for all but a negligible fraction of $\{0,1\}^{k-1}$.*

It is clear that generator G exists and that our family satisfies the requirements of Section 5.1, and that it therefore supports the construction of the signature scheme Σ^* as indicated in Section 5.3.

Combining this with Theorem 2, we have the following.

Corollary 1 *Let the signature scheme Σ^* from Section 5.3 be instantiated for the family $\mathcal{F}_{fact}$ from Definition 1. Then the signature scheme Σ^* is not existentially forgeable under adaptively chosen message attacks. Let l and d be fixed integers, and let k be the size of the moduli, and let a list consisting of l random $(k-1)$-bit strings be shared by all signers. With respect to a public key of size k bits, at least l^d signatures can be made. The size of a signature is at most $(4d-3)k$ bits.*

7. Comparison

We will now compare our scheme Σ^* with [3] and [1]. It is assumed that our signature scheme Σ^* is implemented according to Section 6.2.

7.1 Storage

In terms of storage for the signer, breadth first generation of the authentication tree may not be an optimal strategy: although the shortest signatures are given away first, the storage needed grows rapidly, leaving the signer to record almost the complete history of the authentication tree.

This holds for [3], [1], as well as our schemes. Unless the nodes are generated using a pseudo-random function, as suggested by Goldreich in the case of [3], depth-first development of the tree seems to be a more efficient approach in this respect, as with an authentication tree of depth at most d, only as much as d nodes have to be stored.

In comparison with [1], a signer in Σ^* will have to store an additional lk bits for the permutation pairs, while sharing the list of roughly l random strings is a common property. Note that, if the list Y in the scheme of [1] would be chosen to be consisting of random exponents (in case the corresponding RSA-assumption seems more appealing than the one where, say, l small primes are chosen), the size of this list would be an expected lk bits.

In an application where the shared lists are stored locally by each signer, the total storage needed in the scheme of [1] is the same as in ours.

7.2 Computation

In all three schemes, generation and verification of signatures is roughly the same per basic authentication step (if the claw-free family is based on factoring)

7.3 Size of Signatures

In signature scheme Σ^*, l^d signatures have size at most $(4d-3)k$ bits, whereas in [1] l^{d+1} signatures have size at most $(2d+1)k$ bits. Setting $k = l = 1000$ and $d = 2$ in [1] and $k = l = 1000$ and $d = 3$ for Σ^*, for instance, more than a billion signatures can

be made in both cases. The size of signatures would be at most 5 Kbits and 9 Kbits respectively. At present, this seems to be the price that has to be paid for constructing secure and small signatures on a general cryptographic assumption potentially weaker than an RSA-assumption.

7.4 Various

Our signature scheme Σ^* allows flexibility in setting up the system in the following sense. Contrary to [1], where the size of the shared list is determined by the security parameter, this size can be set independently of the security parameter in our scheme. This means that the size of signatures and the amount of shared random strings can be traded off against each other while keeping the security parameter fixed.

Another difference between [1] and our scheme is that for the generation and verification of a signature in [1] roughly half the elements in the shared list are needed, i.e., $\frac{l}{2}$ k-bit strings, whereas we only need to retrieve at most d of these elements.

In the example where $k = l = 1000$ and $d = 2$ for [1] and $k = l = 1000$ and $d = 3$ for Σ^*, this means retrieving 500 Kbits in [1], versus 3 Kbits in our scheme. Moreover, in [1],

8. Open Problems

In [4] it was shown that secure and efficient signatures can be based on any problem that allows a *collision intractible interactive protocol*. Moreover, that construction does not require trapdoor properties. Examples of these primitives include factoring, discrete logarithms and RSA. Can trade-offs between shared randomness and the size of signatures, similar to that decribed in this paper and in [1], be achieved in the construction of [4]?

9. Conclusion

In the context of secure signature schemes, we have investigated trade-offs between shared randomness and the size of signatures. This approach was initiated in [1].

Although their secure scheme, exploiting special properties of RSA that seem to extend beyond those of trapdoor permutations, still yields signatures that are twice as short, we have shown that, in their model of randomness shared by all signers, secure signatures schemes where the size of the signatures grows only very slowly, can be obtained under the general cryptographic assumption that a family of claw-free trapdoor permutations exists, which is an assumption weaker than an RSA-assumption, as it is known to hold under the factoring assumption.

10. Acknowledgements

I thank Berry Schoenmakers for many effective discussions regarding both technical matters and presentation of the results. Furthermore, I thank Ivan Damgård, Torben Pedersen and Victor Shoup for useful comments.

References

1. C. Dwork, M. Naor: *An Efficient Existentially Unforgeable Signature Scheme and its Applications*, Proceedings of Crypto'94, August 1994, Springer Verlag LNCS series, pp. 234–246.
2. O. Goldreich: *Two Remarks Concerning the GMR Signature Scheme*, Proceedings of Crypto '86, Springer Verlag LNCS series, pp. 104–110.
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. R. Cramer, I. Damgård: *Secure Signature Schemes based on Interactive Protocols*, BRICS Report Series RS–94–29, Aarhus University, Denmark, September '94.