BRICS

Basic Research in Computer Science

BRICS RS-94-29 Cramer & Damgård: Secure Signature Schemes Based on Interactive Protocols

Secure Signature Schemes Based on Interactive Protocols

Ronald Cramer
Ivan Damgård

BRICS Report Series **RS-94-29**

ISSN 0909-0878 **September 1994**

See back inner page for a list of recent publications in the BRICS Report Series. Copies may be obtained by contacting:

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK - 8000 Aarhus C
Denmark**

**Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@daimi.aau.dk**

Secure Signature Schemes based on Interactive Protocols

Ronald Cramer (CWI, Amsterdam),
Ivan Damgård (BRICS [*], Aarhus University)

September 15, 1994

Abstract

A method is proposed for constructing from interactive protocols digital signature schemes secure against adaptively chosen message attacks. Our main result is that practical secure signature schemes can now also be based on computationally difficult problems other than factoring (see [9]), such as the discrete logarithm problem.

More precisely, given only an interactive protocol of a certain type as a primitive, we can build a (non-interactive) signature scheme that is secure in the strongest sense of Goldwasser, Micali and Rivest (see [9]): not existentially forgeable under adaptively chosen message attacks. There are numerous examples of primitives that satisfy our conditions, e.g. Feige-Fiat-Shamir, Schnorr, Guillou-Quisquater, Okamoto and Brickell-Mc.Curley ([7], [15], [10], [13], [3]).

In fact, the existence of one-way group homomorphisms is a sufficient assumption to support our construction. As we also demonstrate that our construction can be based on claw-free pairs of trapdoor one-way permutations, our results can be viewed as a generalization of [9].

[*]Basic Research in Computer Science, Centre of the Danish National Research Foundation

1 Introduction

This paper deals with the construction of *secure* signature schemes. By "secure", we mean that some well-defined computational assumption can be shown to be sufficient for the scheme not to be existentially forgeable, even under an adaptive chosen message attack. This notion of optimal security was introduced in [9]. Most, if not all, signature schemes used in practice such as ISO9796/RSA or DSA are based on a computational assumption that is certainly *necessary* for this kind of security, but not known to be *sufficient*.

Goldwasser, Micali and Rivest [9] were the first to find a provably secure signature scheme, based on the existence of claw-free pairs of trapdoor one-way permutations. Merkle [11] showed essentially that existence of collision intractable hash functions is a sufficient assumption. Naor and Yung showed that any one-way permutation is also enough [12], and finally this was reduced to any one-way function (which is also a necessary assumption) by Rompel [14].

Although secure signature schemes are generally less efficient than the ones used in practice, the efficiency of the GMR scheme is not too bad when based on factoring, and by relying on the (perhaps) stronger assumption that RSA is hard to invert, Bos and Chaum [2] have been able to build an even more efficient secure scheme.

Recently, Dwork and Naor [5] have exhibited an efficient and secure signature scheme whose security is also equivalent to the difficulty of RSA-inversion. In contrast with other schemes that use authentication trees, such as [9], they are able to re-use the authenticating nodes many times. As a result of this and further exploitations of the specific properties of the RSA functions, the length of their signatures can be made quite small, although a price has to be paid in the form of a large public file.

On the theoretical side, the reduction in the necessary assumptions by [11], [12] and [14] have come at the price of dramatically reduced efficiency. In particular, signatures have become larger. Where a GMR signature is of length $O(k)$ bits, where k is the security parameter (ignoring here any dependency on the number of messages signed), a Naor-Yung signature would typically be of length $O(k^2)$ bits, because a full preimage under a one-way function is required to authenticate 1 bit.

Thus it has been an open question whether secure signatures with efficiency comparable to or better than that of GMR could be based on more general assumptions than claw-free pairs of trapdoor one-way permutations.

In this paper, we show that secure signature schemes with signatures as short as those of GMR can be built if so called signature protocols exist. In particular, our schemes have the same property as GMR that the length of signatures grow logarithmically with the number of messages signed. Note that Goldreich [8] has shown that the GMR scheme can be modified so that *all* signatures have length $O(k \log k)$ bits. This same modification applies to our scheme as well.

Dropping some technical details, a signature protocol is an interactive protocol for a hard problem that uses three messages, where the prover speaks first and the verifier sends a random challenge as the second message. The essential properties are

- The protocol must be secure (zero-knowledge) against the honest verifier.
- The challenge must be longer than the prover's first message.
- It must be infeasible for a cheating prover to answer more than one challenge in a given protocol execution.

We show that it is sufficient for the existence of signature protocols that one-way group homomorphisms exist. This has a nice theoretical consequence, because it shows that, compared to GMR, the trapdoor property can be traded for the homomorphism property without getting longer signatures. Moreover, our construction allows us, in both signature generation and verification, to minimize the number of evaluations of the one-way function and replace them by evaluations of the group operation in the the groups involved. This means that we can use the discrete logarithm assumption as a basis for secure signatures in a much more efficient way than known before. Where earlier methods would, with security parameter k, require $O(k^2)$ exponentiations per basic authentication step and give signatures of length $O(k^2)$ bits, our method requires $O(1)$ exponentiations and gives signatures of length $O(k)$.

We also show that existence of a three pass public coin proof of knowledge for any hard problem [1] and a collision intractable hash function implies existence of signature protocols. Although the hash function alone would be

[1]A hard random self-reducible problem would be enough for this

$$
\begin{array}{lcr}
P & & V \\
(a, \mathrm{aux}(a)) \leftarrow P_a(x, w) & & \\
 & \xrightarrow{\quad a \quad} & \\
 & & c \leftarrow \{0,1\}^{C_{\mathcal{P}}} \\
 & \xleftarrow{\quad c \quad} & \\
r \leftarrow P_r(x, w; a, \mathrm{aux}(a), c) & & \\
 & \xrightarrow{\quad r \quad} & \\
 & & \phi(x, a, c, r) \stackrel{?}{=} 1
\end{array}
$$

Figure 1: Protocol $\mathcal{P}$, common input x, private input for P is w

enough to construct secure signatures, using our method may lead to shorter signatures ($O(k)$ compared to $O(k^2)$), depending on the protocol used.

2 Signature Protocols

This section is devoted to defining the basic building block, a *signature protocol*, that is used in our construction for secure signatures.

Let $\mathcal{P}$ be a three round public coin protocol where the prover speaks first. Figure 1 depicts the kind of protocol we will look at. It resembles a proof of knowlege for a binary relation R (see for instance [6] for details), in that the prover can always make the verifier accept on common input x, if the prover knows w such that $(x, w) \in R$.

Indeed, by running (probabilistic) polynomial time algorithm P_a on x and his secret witness w, the prover P computes his initial message a, and some (secret) auxiliary information $\mathrm{aux}(a)$. The length of this first message a is denoted $A_{\mathcal{P}}$, the *authentication length*, which only depends on x. After having received a, the verifier V chooses a challenge c uniformly at random, and sends it to P. The length of admissible challenges in $\mathcal{P}$ is called the *challenge length* $C_{\mathcal{P}}$ (we will sometimes abuse this notation to refer to the *set* of possible challenges). Also here, it is assumed to depend only on x. The prover P completes the conversation by running (probabilistic) polynomial time algorithm P_r on x, w, a, c, and, the auxiliary information $\mathrm{aux}(a)$ for

a. The resulting response r is submitted to the verifier V. We will assume that the procedure ϕ that the verifier V invokes to test the validity of the conversation, is a polynomial time algorithm. The collection of all possible accepting conversations with respect to x will be denoted $Acc(x)$. For the rest of this paper, $\mathcal{P}$ will denote a protocol as described above.

For the purpose of constructing secure signature schemes, the protocol $\mathcal{P}$ does not, however, have to satisfy the ordinary soundness condition. Instead, we require the following.

Definition 1 *Let k be a security parameter for protocol $\mathcal{P}$. Suppose we are given a probabilistic polynomial time generator G for relation R that on input 1^k produces $(x, w) \in R$, such that no probabilistic polynomial time algorithm, given x as input, can generate two accepting conversations (with respect to x) (a, c, r), (a, c', r') from $Acc(x)$, with $c \neq c'$, except with negligible probability of success. Then $\mathcal{P}$ is called* collision intractible *over G.*

Next, we need the protocol $\mathcal{P}$ to be secure in the following sense. Instead of requiring the protocol to be zero-knowledge against an arbitrary verifier, we only demand that conversations with an *honest verifier* (i.e., a verifier who follows protocol $\mathcal{P}$ as desired) can be simulated. Additionally, we require that the simulator outputs accepting conversations where the challenge can be chosen in advance, i.e., the simulator can take any value c as input, and will output an accepting conversation where the challenge is equal to c. A protocol $\mathcal{P}$ satisfying these conditions will be called *special honest verifier zero-knowledge.*

More precisely, let $(x, w) \in R$ and let a prover P and a verifier V with common input x be given. The prover has w as private input. Then $\mathcal{P}(x, w)$ denotes the probability distribution on $Acc(x)$ induced by conversations between P and V, provided that they both follow protocol $\mathcal{P}$ honestly. We require the following.

Definition 2 *Let $(x, w) \in R$. Suppose we are given a probabilistic polynomial time algorithm $\mathcal{S}$ with the following properties.*

1. *On input x and any $c \in C_{\mathcal{P}}$, $\mathcal{S}$ outputs an accepting conversation from $Acc(x)$.*

2. *The distribution of $\mathcal{S}(x, c)$, where c is chosen uniformly at random from $C_{\mathcal{P}}$, is equal to $\mathcal{P}(x, w)$.*

Then $\mathcal{P}$ is called special honest verifier zero knowledge*, and $\mathcal{S}$ its* special simulator.

In the following we will demonstrate that a protocol $\mathcal{P}$ that is special honest verifier zero-knowledge, is in fact secure against a slightly more general verifier. It follows immediately from Definition 2 that, for each fixed $c \in C_{\mathcal{P}}$, $\mathcal{S}(x, c)$ outputs conversations $(a, c, r) \in Acc(x)$ with exactly the same distribution as $(a \leftarrow P_a(x, w), c, r \leftarrow P_r(x, w, a, \text{aux}(a), c))$, i.e., according to the honest prover who has access to (x, w). Therefore, it is sufficient that challenges c are independently chosen from the first message in any given execution of $\mathcal{P}$, in order for the conversations to be simulatible. In other words, c may depend on anything (including the history of executions, x, etc.) but the prover's first message a in the given execution, and the conversation is still simulatible. This proves the following theorem.

Theorem 1 *If $\tilde{V}$ is any probabilistic polynomial time verifier who, in any given execution of protocol $\mathcal{P}$, chooses the challenge c independently from the prover's first message a, then the conversation between prover P and verifier $\tilde{V}$ can be simulated by means of the special simulator $\mathcal{S}$.*

Summarizing, we require the following of our protocol $\mathcal{P}$ in order for it to support our construction of (non-interactive) secure signature schemes.

Definition 3 *Suppose $\mathcal{P}$ satisfies the following conditions.*

1. *$C_{\mathcal{P}} > A_{\mathcal{P}}$.*
2. *$\mathcal{P}$ is collision-intractible over G.*
3. *$\mathcal{P}$ is special honest verifier zero-knowledge.*

Then $\mathcal{P}$ is called a signature protocol*. If $\mathcal{P}$ satisfies the second condition and is honest verifier zero-knowledge (so it does not necessarily have a special simulator), $\mathcal{P}$ is called a* quasi signature protocol.

We now demonstrate that any given signature protocol $\mathcal{P}$ can be transformed into a new signature protocol $\mathcal{P}^*$ where the challenge length $C_{\mathcal{P}^*}$ can be of any size polynomial in the security parameter k.

Theorem 2 *Suppose there exists a signature protocol $\mathcal{P}$ for relation R and generator G, then there is a signature protocol $\mathcal{P}^*$ for R and G, satisfying that $C_{\mathcal{P}^*} = t$, for any t polynomial in the security parameter k.*

Proof: Without loss of generality, we may assume that $A_{\mathcal{P}} + 1 = C_{\mathcal{P}}$ The protocol $\mathcal{P}^*$ goes as follows:

1. The prover sends a first message a to the verifier, where a is computed as in $\mathcal{P}$.
2. The verifier sends t random bits $b_1, \dots, b_t$.
3. The prover sends t conversations in $\mathcal{P}$, $(a_i, c_i, r_i), i = 1, \dots, t$, where $c_i = b_i || a_{i+1}$ for $i = 1, \dots, t-1$ and $c_t = b_t || 0 || \cdots || 0$.
4. The verifier checks that $a = a_1$, that all conversations are accepting conversations, and that $c_i = b_i || a_{i+1}$ for $i = 1, \dots, t-1$, and that $c_t = b_t || 0 || \cdots || 0$.

By construction, the challenge length t for $\mathcal{P}^*$ can be chosen what we want it to be, provided $t = \text{poly}(k)$. Suppose now that we are given two accepting conversations in $\mathcal{P}^*$ for some public string x with the same first message a, but with different challenges $(b_1, \dots, b_t)$ and $(b'_1, \dots, b'_t)$. Let, for $j = 1 \dots t$, (a_j, c_j, r_j) and (a'_j, c'_j, r'_j) be the respective replies in those conversations in $\mathcal{P}^*$, and let i be an index such that $b_i \neq b'_i$. Clearly, this implies that $c_i \neq c'_i$. Take i to be the smallest index such that $c_i \neq c'_i$. If $i = 1$, we have a collision in $\mathcal{P}$ with respect to x, as by definition of $\mathcal{P}^*$, we must have $a_1 = a'_1 = a$.On the other hand, if $i > 1$, c_{i-1} must be equal to c'_{i-1}, i.e., $b_{i-1} || a_i = b'_{i-1} || a'_i$. But then $a_i = a'_i$ and we have a collision (a_i, c_i, r_i), (a'_i, c'_i, r'_i) in $\mathcal{P}$ with respect to x. Therefore, $\mathcal{P}^*$ is collision-intractible over R and G.

As for special honest verifier zero-knowledge of $\mathcal{P}^*$, we now exhibit a special simulator $\mathcal{S}^*$ for $\mathcal{P}^*$, that runs $\mathcal{S}$ as a subroutine. $\mathcal{S}^*$ starts by receiving a public string x and a challenge $(b_1, \dots, b_t)$ as input. It proceeds by putting $c_t = b_t || 0 || \cdots || 0$, and feeding x and c_t to $\mathcal{S}$. After $\mathcal{S}$ has output an accepting conversation (a_t, c_t, r_t) in $\mathcal{P}$ with respect to x, $\mathcal{S}^*$ repeats the following for $i = t-1 \dots 1$. Put $c_i = b_i || a_{i+1}$, feed x and c_i to $\mathcal{S}$ and receive an accepting conversation (a_i, c_i, r_i) from $\mathcal{S}$. By invoking Theorem 1, it is clear that $\mathcal{S}^*$ generates accepting conversations in $\mathcal{P}^*$ with respect to x, with exactly the same distribution as the conversations with the honest verifier in $\mathcal{P}^*$. □

Thus, in the constructions to follow, whenever we have a signature protocol, we may assume that the challenge length is whatever we need it to be.

Before investigating under which general assumptions signature protocols can be shown to exist, we mention some examples of proofs of knowledge that can be viewed as signature protocols.

- Guillou-Quisquater [10].
- Okamoto [13], both the factoring and the RSA-versions.
- Fiat-Shamir [7] (if the number of secret roots is chosen sufficiently large)

Schnorr's discrete log protocol [15] does not directly satisfy the conditions, but can be modified to do so since it is based on a one-way group homomorphism (see below).

3 Sufficient Assumptions

The most general computational assumptions we have been able to find, sufficient for existence of signature protocols, is the existence of one-way group homomorphisms, and the existence of claw-free pairs of trapdoor one-way permutations. No implication is known in either direction between these two assumptions.

One-Way Group Homomorphisms

Definition 4 *A family of* one-way group homomorphisms *is a family of group homomorphisms $\mathcal{F} = \{f : G \to H\}$. In the following, we let $k_f = \log_2(|H|)$, i.e. the number of bits needed to represent an element in H. We will sometimes drop subscript f, if it is clear which f we refer to. The family has to satisfy the following properties:*

1. *There is a polynomial time algorithm which given f and $w \in G$, computes $f(w)$ in time polynomial in k.*

2. *There is a probabilistic polynomial time algorithm which on input 1^k outputs an element $f : G \to H$ chosen uniformly from $\mathcal{F}$, subject to $k = k_f$.*

3. *The elements $f : G \to H \in \mathcal{F}$ satisfy that there is a probabilistic algorithm which given G outputs an element chosen uniformly from G, in time polynomial in k.*

4. *The one-way property: Let A be any probabilistic polynomial time algorithm which receives input f and $f(w)$, where f, w are chosen as in points 2 and 3. Then the probability that A outputs y such that $f(y) = f(w)$ is superpolynomially small in k.*

5. *The elements $f : G \to H \in \mathcal{F}$ satisfy that group operation and inversion in G and H can be computed in time polynomial in k.*

An example of such a family could be the case where the homomorphisms are discrete exponentiation modulo a prime, i.e. each element $f : G \to H$ is described by a k-bit prime p and an element $g \in Z_p$. G is the additive group modulo $p-1$, H is the multiplicative group modulo p, and $f(w) = g^w \bmod p$.

Given a family as in this definition, we can make a binary relation and a generator for it:

Definition 5 *Let $\mathcal{F}$ be as in Definition 4. Then $R_{\mathcal{F}}$ is the binary relation consisting of pairs $((f, x_1, \ldots, x_{k_f+1}), (w_1, \ldots, w_{k_f+1}))$, where $f \in \mathcal{F}$ and $f(w_i) = x_i$. $G_{\mathcal{F}}$ is the generator that on input 1^k generates f using property 2 of Definition 4, generates $w_1, \ldots, w_{k_f+1}$ using property 3 and finally computes $x_i = f(w_i)$.*

Theorem 3 *Suppose $\mathcal{F}$ is a family of one-way group homomorphisms. Then there exists a signature protocol for $R_{\mathcal{F}}$ and $G_{\mathcal{F}}$.*

Proof: The protocol claimed takes $f, x_1, \ldots, x_{k+1}$ as common input, while $w_1, \ldots, w_{k+1}$ are private input to the prover. The protocol is now a straightforward generalization of Feige-Fiat-Shamir [7] and goes as follows:

1. The prover chooses a random $r \in G$ and sends $f(r)$ to the verifier.
2. The verifier chooses bits $e_1, \ldots, e_{k+1}$ at random and sends them to the prover.

3. The prover returns $z = r \cdot w_1^{e_1} \cdots w_{k+1}^{e_{k+1}}$. The verifier checks that $f(z) = f(r) \cdot x_1^{e_1} \cdots x_{k+1}^{e_{k+1}}$

This protocol is clearly complete with probability 1. Honest verifier zero knowledge is clear by standard arguments: first choose z and $e_1, \ldots, e_{k+1}$ at random, then use this to compute an $f(r)$-value. It is also clear that the challenge is one bit longer than the first message from the prover. Thus, only the collision intractable property remains to be argued:

So assume by contradiction that some enemy A can produce z, z' and $(e_1, \ldots, e_{k+1}) \neq (e'_1, \ldots, e'_{k+1})$ such that $f(z) = f(r) \cdot x_1^{e_1} \cdots x_{k+1}^{e_{k+1}}$ and $f(z') = f(r) \cdot x_1^{e'_1} \cdots x_{k+1}^{e'_{k+1}}$. This means that

$$f(z \cdot z'^{-1}) = x_1^{d_1} \cdots x_{k+1}^{d_{k+1}},$$

where all d_i are 1, -1 or 0, and at least one of them is non-zero.

We can then build the following algorithm which will invert f with the help of A: given a random f-image x, generate an output seemingly coming from $G_{\mathcal{F}}$ as follows: choose $w_1, \ldots, w_{k+1}$ and $1 \leq j \leq k+1$ at random. Put $x_i = f(w_i)$ for $i \neq j$, and $x_j = f(w_j) \cdot x$. Now run A's algorithm with f and the x_i's as input. Clearly the set of x_i is distributed exactly as output from $G_{\mathcal{F}}$, whence A's success probability is the same as in real life. Note that if A has success, we can write x^{d_j} as

$$x^{d_j} = f(z \cdot z'^{-1} \cdot \prod_i w_i^{-d_i})$$

Now note that the set of x_i's contains no information about j, whence the probability that $d_j \neq 0$, given that A has success, is at least equal to $1/(k+1)$. □

Remark 1 *It is clear that the protocol constructed in the proof above can be modified to have any challenge length desired by having more x_i-values. Enlarging the challenge length in this way will be more efficient than using Theorem 2.*

Examples of possible one-way group homomorphisms are the RSA functions, squaring modulo a composite number, or discrete exponentiation modulo a prime, or on an elliptic curve.

It would be natural to try to generalize the result to any random self-reducible problem. It is known that a random self-reducible problem has a protocol that is in our terminology a quasi-signature protocol [16]. It is not clear, however, how to get longer challenges based only on the self-reducible property. But if in addition we assume we have a family of collision intractable hash functions we can use the compression properties of the hash functions to build a signature protocol. Briefly, a family of collision intractable hash functions $\mathcal{H}$ is a family of easily computable compression functions, such that it is easy to select a random function with output length k but computationally infeasible to find collisions for such a function with probability non-negligible in k.

Since, however, quasi-signature protocols are not assumed to be special honest verifier zero-knowledge (only honest verifier zero-knowledge), we need the following technical lemma before going any further.

Lemma 1 *Let $\mathcal{P}$ be honest verifier zero-knowledge and collision-intractable over R and G. Then $\mathcal{P}$ can be compiled into a protocol $\mathcal{P}^*$ (for relation R and generator G), that is also collision-intractable over R and G but that additionally satisfies* special *honest verifier zero-knowledge.*

Proof: The claimed protocol works as follows. The prover has access to $(x, w) \in R$, while the verifier has access to x. Let k be a security parameter, let $l = \text{poly}(k)$, and let $\{0,1\}^t$ be the set of admissible challenges in $\mathcal{P}$.

1. The prover computes l first messages $a_1, \ldots, a_l$ as in $\mathcal{P}$, and sends them to the verifier.
2. The verifier chooses l random bits $b_1, \ldots, b_l$, and sends them to the prover.
3. The prover chooses l random $t-1$-bitstrings $\beta_1, \ldots, \beta_l$, and puts $c_1 = \beta_1 || b_1, \ldots, c_l = \beta_1 || b_l$, and computes the responses $r_1, \ldots, r_l$ according to $\mathcal{P}$, taking the challenges to be $c_1, \ldots, c_l$. These values are sent to the verifier, who checks whether $c_1^{(t)} = b_1, \ldots, c_l^{(t)} = b_l$ and whether $(a_1, c_1, r_1), \ldots, (a_l, c_l, r_l)$ are accepting conversations in $\mathcal{P}$ with respect to x.

First, we show that collision-intractibility is preserved under this compilation. Suppose we are given two accepting conversations in $\mathcal{P}^*$, with the same first message $(a_1, \ldots, a_l)$, but with different challenges $(b_1, \ldots, b_l)$ and $(b'_1, \ldots, b'_l)$. Let the respective replies be $(\beta_1, \ldots, \beta_l, r_1, \ldots, r_l)$ and $(\beta'_1, \ldots, \beta'_l, r'_1, \ldots, r'_l)$, and let i be an index such that $b_i \neq b'_i$. Then clearly, $(a_i, \beta_i || b_i, r_i)$ and $(a_i, \beta'_i || b'_i, r'_i)$ are two accepting conversations in $\mathcal{P}$ for the same public string x, with $\beta_i || b_i \neq \beta'_i || b'_i$. We conclude that $\mathcal{P}^*$ is collision-intractable over R and G.

The special simulator $\mathcal{S}^*$ for $\mathcal{P}^*$ runs $\mathcal{P}$'s simulator $\mathcal{S}$ as a subroutine, and is defined as follows. Run $\mathcal{S}$ $2l$ times. At the end, there are certainly l conversations that have challenges with the same least significant bit, "0" or "1". By the properties of $\mathcal{S}$, these events are equally likely to occur. If we repeat this procedure poly(k) times, the probability that all resulting blocks of l conversations have the same "parity" is equal to $\frac{1}{2}^{\text{poly}(k)}$. So, with overwhelming probability two blocks of l conversations are output, one of which has "0" as the least significant bit for all its l conversations, while the other has "1". Therefore, if $\mathcal{S}^*$ receives a challenge $(b_1, \ldots, b_l) \in \{0,1\}^l$ as input, together with the public string x, it can output an accepting conversation (in $\mathcal{P}^*$) with $(b_1, \ldots, b_l)$ as the challenge in polynomial time with overwhelming probability, by just selecting, for each b_i, a conversation from the corresponding block. Furthermore, it is clear that the honest verifier in $\mathcal{P}^*$ receives l conversations from $\mathcal{P}$ where each of these conversations is according to conversations with an honest verifier in $\mathcal{P}$. By construction, it is clear that $\mathcal{S}^*$ does the same: using simulator $\mathcal{S}$ to select honest verifier conversations in $\mathcal{P}$ according to the least significant bit in the challenge, while the selection is according to uniform bits. □

Theorem 4 *Suppose there exists a quasi signature protocol $\mathcal{P}$ for relation R and generator G and that a family $\mathcal{H}$ of collision intractable hash functions exists. Then there exists a signature protocol $\mathcal{P}^*$ for $R_{\mathcal{H}}$ and $G_{\mathcal{H}}$. Here $R_{\mathcal{H}}$ consists of pairs $((x,h),w)$ where $(x,w) \in R$, w is of length k bits and $h \in \mathcal{H}$ has output length k. The generator $G_{\mathcal{H}}$ runs G to generate (x,w) and then selects $h \in \mathcal{H}$ with the desired output length.*

Proof: First note that by Lemma 1, we may assume that $\mathcal{P}$ is special honest verifier zero-knowledge. Then observe that a repetition of $\mathcal{P}$ in parallel is trivially a quasi signature protocol. Moreover, from any quasi signature protocol, we can always construct a new one with any smaller challenge length by letting the prover choose part of the challenge. Hence we may without loss of generality assume that $C_{\mathcal{P}} = k + 1$. Let $t = A_{\mathcal{P}} + 1$. Then protocol $\mathcal{P}^*$ goes as follows:

1. The prover sends a first message a to the verifier computed as in $\mathcal{P}$.
2. The verifier sends t random bits $b_1, \ldots, b_t$.
3. The prover sends t conversations in $\mathcal{P}$, $(a_i, c_i, r_i), i = 1, \ldots, t$.
4. The verifier checks that $a = a_1$, that all conversations are accepting conversations, and that $c_i = b_i || h(a_{i+1})$ for $i = 1, \ldots, t-1$, and that $c_t = b_t || 0 || \cdots || 0$.

It is easy to verify that this protocol has all the required properties. (see the proof of Theorem 2). In particular, collision intractability can be proved observing that a collision for $\mathcal{P}^*$ would imply either a collision for $\mathcal{P}$ or for h. □

We have chosen to use in the above theorem a whole family of hash functions (in stead of a single fixed function) because this fits into our theoretical model. In practice, many hash functions do not come from a family but have a fixed description, such as MD4 or SHS. Our construction will also work with one fixed hash function, and the argument that a successful enemy would have to break either the hash function or the quasi signature protocol would be the same as before.

Claw-Free Pairs of Trapdoor One-Way Permutations

In [9], a secure signature scheme is exhibited, based on (a family of) claw-free pairs of trapdoor one-way permutations. Informally, a pair of distinct permutations (f_0, f_1) is called claw-free, if it is hard to compute x and y such that $f_0(x) = f_1(y)$. Knowledge of the trapdoor information, however, enables efficient inversion of the permutations and computation of claws. In [9], an example of such a family is given, whose claw-freeness is equivalent to the difficulty of factoring Blum-integers.

In the following we will show that the existence of a family of claw-free pairs of trapdoor one-way permutations is a sufficient condition for the existence of signature protocols. Moreover, building a signature protocol from a

P		V
$a \leftarrow V$		
	$\xrightarrow{\quad a \quad}$	
		$c \leftarrow \{0,1\}^t$
	$\xleftarrow{\quad c \quad}$	
$r \leftarrow f_{[c]}^{-1}(a)$		
	$\xrightarrow{\quad r \quad}$	
		$a \stackrel{?}{=} f_{[c]}(r)$

Figure 2: Signature Protocol $\mathcal{P}$ based on (f_0, f_1)

claw-free pair of trapdoor permutations as described below and then applying our general construction to this signature protocol results in essentially the same signature scheme as the original GMR sheme. Hence our results can be viewed as a generalization of [9].

Let (f_0, f_1) be a pair of functions from a family $\mathcal{F}$ of claw-free pairs of trapdoor one-way permutations as output by a generator $G_{\mathcal{F}}$ on input 1^k, and let s denote the trapdoor information. The corresponding binary relation $R_{\mathcal{F}}$ consists of all such pairs $((f_0, f_1), s)$.

Furthermore, let t be a non-constant polynomial in k. For each $c \in \{0,1\}^t$, with $c = c_1||c_1||\cdots||c_t$, $f_{[c]}$ denotes the function $f_{c_1} \circ f_{c_1} \circ \cdots \circ f_{c_t}$, which is also a permutation. Using the trapdoor information for (f_0, f_1), such a function $f_{[c]}$ can efficiently be inverted. Note also that any pair of permutations $(f_{[c]}, f_{[c']})$, with $c, c' \in \{0,1\}^t$ and $c \neq c'$, is claw-free. Let V denote the set that is permuted by f_0 and f_1. The protocol $\mathcal{P}$, depicted in Fig. 2 is based on (f_0, f_1), and it is assumed that the prover P has access to the trapdoor information s.

This protocol satisfies the conditions of a signature protocol by standard arguments. We thus have the following theorem.

Theorem 5 *Suppose $\mathcal{F}$ is a family of claw-free pairs of trapdoor one-way permutations. Then there exists a signature protocol for $R_{\mathcal{F}}$ and $G_{\mathcal{F}}$.*

4 Main Result

We will now present the new signature scheme $\Sigma_{\mathcal{P}}$, based on a signature protocol $\mathcal{P}$. In Section 5, the following theorem will be proven.

Theorem 6 *Let $\mathcal{P}$ be a signature protocol for relation R and generator G. Then the signature scheme $\Sigma_{\mathcal{P}}$ is not existentially forgeable under adaptively chosen message attacks.*

It is assumed that we are given a signature protocol $\mathcal{P}$ for relation R and generator G. By Theorem 2, we may assume that for each security parameter k and for each instance (x, w) as output by running $G(1^k)$, the (non-constant) polynomial $t(k)$ satisfies $t = C_{\mathcal{P}} \geq 3 \cdot A_{\mathcal{P}}$. The construction of $\Sigma_{\mathcal{P}}$ from $\mathcal{P}$ works as follows.

Initialization Phase

Given a security parameter k, the signer uses the generator G to generate two solved instances x_0 and x_1, with respective witnesses w_0 and w_1. He also computes $(a_1^1, \mathrm{aux}(a_1^1)) \leftarrow P_a(x_1, w_1)$ and puts (x_0, x_1, a_1^1) in his public directory.

Signing Phase

Let $m \in \{0,1\}^t$ be the message to be signed and let $i \geq 1$. The i-th signature, on a message $m \in \{0,1\}^t$, is computed as follows. First, the signer computes

1. $(a_0^i, \mathrm{aux}(a_0^i)) \leftarrow P_a(x_0, w_0)$,
2. $r_0^i \leftarrow P_r(x_0, w_0; a_0^i, \mathrm{aux}(a_0^i), m)$,
3. $(a_1^{2i}, \mathrm{aux}(a_1^{2i})) \leftarrow P_a(x_1, w_1)$, $(a_1^{2i+1}, \mathrm{aux}(a_1^{2i+1})) \leftarrow P_a(x_1, w_1)$,
4. $r_1^i \leftarrow P_r(x_1, w_1; a_1^i, \mathrm{aux}(a_1^i), a_1^{2i} || a_1^{2i+1} || a_0^i)$.

The signer stores a_1^{2i}, $\mathrm{aux}(a_1^{2i})$, a_1^{2i+1}, $\mathrm{aux}(a_1^{2i+1})$, a_0^i, r_1^i. Let $\mathrm{Auth}(a_0^i)$ be an authentication path for a_0^i, i.e. $\mathrm{Auth}(a_0^i)$ consists of all tuples $(a_1^j, a_1^{2j}, a_1^{2j+1}, a_0^j, r_1^j)$, with $1 \leq j \leq i$, such that a_1^j is an ancestor of a_1^i. We assume that the tuples in $\mathrm{Auth}(a_0^i)$ are ordered in decreasing ancestry from left to right. The signature $\sigma(m)$ on m consists of $(\mathrm{Auth}(a_0^i), r_0^i)$.

Verification Phase

The receiver puts $\sigma(m) \equiv (\mathrm{Auth}(a_0^{j_r}), r_0^{j_r})$, where r is the number of tuples in $\mathrm{Auth}(a_0^i)$ and $(a_1^{j_l}, a_1^{2j_l}, a_1^{2j_l+1}, a_0^{j_l}, r_1^{j_l})$ is the l-th tuple in $\mathrm{Auth}(a_0^{j_r})$. After having checked whether $a_1^{j_1} \stackrel{?}{=} a_1^1$, the receiver has to perform the following verifications, for $j = 2, \ldots, r$.

1. $a_1^{j_l} \stackrel{?}{\in} \{a_1^{2j_{l-1}}, a_1^{2j_{l-1}+1}\}$
2. $\phi(x_1, a_1^{j_l}, a_1^{2j_l}||a_1^{2j_l+1}||a_0^{j_l}, r_1^{j_l}) \stackrel{?}{=} 1$.

Finally, he checks whether $\phi(x_0, a_0^{j_r}, m, r_0^{j_r}) = 1$. If all verifications hold, the signature is accepted.

Note that, by assumption on the challenge length $t(k)$, $2 \cdot A_{\mathcal{P}}(x_1) + A_{\mathcal{P}}(x_0) \leq t$, so the challenges are long enough to encode the strings $a_1^{2i}||a_1^{2i+1}||a_0^i$. These strings can be padded up to t bits, if necessary, using standard techniques. As we have also assumed that all occurring values have fixed length descriptions (depending only on the corresponding public string), parsing these concatenations is easy.

5 Proof of Security

Our notion of security for signature schemes is that of [9]. In this section we show that no polynomially bounded adversary can construct a forgery on a message that hasn't been signed by the real signer, even if he is allowed to get polynomially many signatures on messages that he has chosen in an adaptive fashion. We first briefly outline the proof of Theorem 6. It will be shown that the existence of such a successful forger contradicts the assumption that the protocol $\mathcal{P}$ is collision intractable over the generator G. To this end, we compile this successful forger into an attacker that breaks that assumption.

Before proceeding with the proof, we will briefly outline our approach. Let k be a given security parameter. A key-observation is that, for any fixed polynomial number, say $P(k)$, of signatures, the signature scheme $\Sigma_{\mathcal{P}}$ can be simulated perfectly and efficiently if one of the two witnesses w_0 and w_1 is discarded right after generation.

Bearing this in mind, we will build a cracking algorithm $\mathcal{A}^*$ which gets a problem instance x (as generated by G) as input, and generates a collision for this instance using the forgery algorithm $\mathcal{A}$ as a subroutine. To do this, $\mathcal{A}^*$ builds an instance of $\Sigma_{\mathcal{P}}$ from x and a pair (x', w') generated by running G. The public key will be the pair (x, x'), randomly permuted. By the perfectness of this simulation of $\Sigma_{\mathcal{P}}$, we can run $\mathcal{A}$ and handle all its signature requests and expect the same probability of success as in "real life". The proof is then finalized by observing that a successsful forgery leads to a collision for the instance x with probability $1/2$.

In the following theorem, it is assumed that we are given a signature protocol $\mathcal{P}$ for generator G and relation R. By Theorem 2, we may assume that for each security parameter k and for each instance (x, w) as output by running $G(1^k)$, the (non-constant) polynomial $t(k)$ satisifies $t = C_{\mathcal{P}} \geq 3 \cdot A_{\mathcal{P}}$.

Theorem 7 *Any probabilistic polynomial time cracking algorithm $\mathcal{A}$ that forges a signature on a new message with probability $\epsilon(k)$, after at most polynomially many calls to a signer, can be compiled into probabilistic polynomial time procedure $\mathcal{A}^*$ that breaks the collision intractability of $\mathcal{P}$ over G with probability of the order of $\epsilon(k)$). The running time of $\mathcal{A}^*$ is of the same order as the running time of $\mathcal{A}$.*

Proof: Let a security parameter k be given, and let x be an instance of $\mathcal{P}$ generated by G on input 1^k.

We now describe how $\mathcal{A}^*$ cracks the collision intractability of $\mathcal{P}$ by using the forger $\mathcal{A}$ and the following simulation of $\Sigma_{\mathcal{P}}$. $\mathcal{A}^*$ receives x as input.

$\mathcal{A}^*$ first runs G on input 1^k in order to obtain a solved instance (x', w'). Then a bit b is chosen at random. Put $(x_b, w_b) = (x', w')$, and $x_{1-b} = x$.

For the simulation, we distinguish between two cases.
Case $b = 0$: We create an authentication tree with $P(k)$ internal nodes, starting at the leaves. The leaves a_1^j are generated as follows.

1. $c^j \leftarrow \{0,1\}^t$
2. $(a_1^j, c^j, r_1^j) \leftarrow \mathcal{S}(x_1, c^j)$.

For children a_1^{2i} and a_1^{2i+1}, generate $a_0^i \leftarrow P_a(x_0, w_0)$. Then the parent a_1^i will be generated as

$$(a_1^i, a_1^{2i}||a_1^{2i+1}||a_0^i, r_1^i) \leftarrow \mathcal{S}(x_1, a_1^{2i}||a_1^{2i+1}||a_0^i).$$

The resulting instance (x_0, x_1, a_1^1) of $\Sigma_{\mathcal{P}}$ is sent to the forger $\mathcal{A}$. After this, the cracking algorithm can start making its (at most $P(k)$) calls.

The above takes care of $\mathrm{Auth}(a_0^i)$, for $i = 1, \ldots, P(k)$. Note that this simulation can now deal with any signature request, as the i-th signature request, on a message m^i, can be completed by computing $r_0^i \leftarrow P_r(x_0, w_0; a_0^i, \mathrm{aux}(a_0^i), m^i)$.

Case $b = 1$**:**

1. Generate $(a_1^1, \mathrm{aux}(a_1^1)) \leftarrow P_a(x_1, w_1)$, and send the instance (x_0, x_1, a_1^1) to the forger $\mathcal{A}$.
2. Let $m^i \in \{0,1\}^t$ be the i-th message to be signed. Generate $a_0^i \leftarrow \mathcal{S}(x_0, m^i)$. Proceed as in Step 3 of the signing phase of $\Sigma_{\mathcal{P}}$.

Note that in both cases the simulation can deal with any signature request, by the properties of the special simulator $\mathcal{S}$. Furthermore, the distribution of the a_0^i, r_0^i, a_1^i and r_1^i is always according to the honest signer who has access to both w_0 and w_1, by Theorem 1. Thus the simulation is perfect, and we may now assume that the cracking algorithm outputs a forgery on a new message (i.e, a message that has not been signed by the simulator) $\tilde{m}$. Without loss of generality, we assume that this happens after exactly $P(k)$ calls, with probability $\epsilon(k)$.

Let $(\mathrm{Auth}(a_0), r_0)$ be the forgery, on a new message $\tilde{m}$. Suppose that $a_0 = a_0^j$ for some $1 \leq j \leq P(k)$, with probability $\epsilon_1(k)$. As $\tilde{m}$ has not been signed by the simulation, we must have $\tilde{m} \neq m^j$, so $\mathcal{A}^*$ can get a collision for $\mathcal{P}$ from $(a_0, \tilde{m}, r_0)$ and (a_0^j, m^j, r_0^j).

If, on the contrary, $a_0 \neq a_0^j$ for all $1 \leq j \leq P(k)$, then there clearly exist a tuple $(a_1', a_1'', a_1''', a_0', r_1')$ in $\mathrm{Auth}(a_0)$ and a node a_1^i in the tree, with $a_1' = a_1^i$, such that a_1^i is a leaf or a_1^i is an internal node with $a_1''||a_1'''||a_0' \neq a_1^{2i}||a_1^{2i+1}||a_0^i$.

In case a_1^i is an internal node, say with probability $\epsilon_2(k)$, we immediately get a collision. If a_1^i is a leaf, with probability $\epsilon_3(k)$, however, the probability that $a_1''||a_1'''||a_0' \neq c^i$ is $1 - \frac{1}{2^t}$, as the distribution of a_1^i is independent of the distribution of c^j (by the properties of the special simulator), and c^j was chosen uniformly at random. Thus in this case we get a collision with probability $1 - \frac{1}{2^t}$. ¿From the perfectness of the simulation it follows that the distribution of everything sent to $\mathcal{A}$ is independent of b. Therefore the probability that $\mathcal{A}^*$ can compute a collision for the instance $x_{1-b} = x$ is

$$\frac{1}{2}\epsilon_1(k) + \frac{1}{2}\epsilon_2(k) + \frac{1}{2}(1 - \frac{1}{2^t})\epsilon_3(k) \geq \frac{1}{2}\epsilon(k) - \frac{1}{2^{t+1}}\epsilon_3(k),$$

which is clearly of the same order as $\epsilon(k)$. Thus we have shown that any forger of the signature scheme $\Sigma_{\mathcal{P}}$ can be turned very efficiently into a cracker of the collision intractibility of $\mathcal{P}$, with essentially the same probability of success.

□

Remark 2 *Consider the following (potentially) stronger notion of security for signature schemes. Instead of requiring forgery on a* new message *to be infeasible, one could, more generally, demand that forging a* new signature *is infeasible. Obviously, this implies that a forger cannot produce a signature on a message that has never really been signed. Additionally, however, it is now infeasible to forge a new signature on a message that has previously been signed. We believe that, from a theoretical viewpoint, this is the proper and most general notion of security for signature schemes. Taking minor changes into account, our proof can easily be accomodated to this (potentially) slightly stronger notion.*

6 Concrete Examples

We now describe a signature scheme whose security is equivalent to the difficulty of computing discrete logarithms, by applying our main construction to a suitable transformation of the discrete log based protocol of Schnorr [15]. In its basic form, this is a protocol for proving knowledge of a discrete log in a group $\mathcal{G}$ of prime order q. Such a group can be realized, for example as a subgroup of $\mathbb{Z}_p^*$, where p is a prime, and q divides $p-1$.

Let $g \neq 1$, and let $x = g^w$ be the common input. P is given w as private input. The protocol is a proof of knowledge for the relation that consists of pairs $((x, g, \mathcal{G}),\ w)$ such that $x = g^w$ in $\mathcal{G}$. Let k denote the number of bits needed to represent an element of $\mathcal{G}$ and let $l = \lfloor \log_2 q \rfloor$. Then the protocol works as follows:

1. The prover chooses z at random in $[0, \ldots, q)$, and sends $a = g^z$ to V.

2. The verifier chooses c at random in $[0, \ldots, q)$, and sends it to P.

3. P sends $r = (z + cw) \bmod q$ to V, and V checks that $g^r = a\,x^c$.

Completeness trivially holds with probability 1. Correct answers to two different c-values give two equations $r_1 = z + wc_1 \bmod q$ and $r_2 = z + wc_2 \bmod q$ so we find that $w = (r_1 - r_2)/(c_1 - c_2) \bmod q$. Therefore, assuming we generate inputs for the protocol by choosing w at random in $\mathcal{G}$, we have collision intractability provided that it is infeasible to find w from g^w for random w.

Finally, note that by choosing c and r at random, we can make a simulated conversation $(g^r x^{-c}, c, r)$ between the honest verifier and prover. Since c can be chosen freely, we get special honest verifier zero-knowledge.

Thus this protocol is a quasi-signature protocol. With some modifications, it can be turned into a signature protocol: we will have as input to the protocol d instances instead of 1, $(x_1, w_1), \ldots, (x_d, w_d)$, where $x_i = g^{w_i}$. Then the new protocol $\mathcal{P}$ goes as follows:

1. The prover chooses z at random in $[0, \ldots, 2^l)$, and sends $a = g^z$ to V.

2. The verifier chooses $c_1, \ldots, c_d$ at random in $[0, \ldots, 2^l)$, and sends them to P.

3. P sends $r = (z + c_1 w_1 + \cdots + c_d w_d) \bmod q$ to V, and V checks that $g^r = a \cdot x_1^{c_1} \cdots x_d^{c_d}$.

Completeness and special honest verifier zero-knowledge are clear by the same arguments as above. Collision intractability can be shown by essentially the same proof as for Theorem 3. Finally, it is clear that by choosing d large enough, we can get a large enough challenge length, and therefore a signature protocol.

We can now carry out our construction of the signature scheme $\Sigma_{\mathcal{P}}$ (see also Section 4). To set up an instance of $\Sigma_{\mathcal{P}}$, the signer generates two independent instances of $\mathcal{P}$, $(x, w) \equiv ((x_1, w_1), \ldots, (x_d, w_d))$ and $(\overline{x}, \overline{w}) \equiv ((\overline{x}_1, \overline{w}_1), \ldots, (\overline{x}_d, \overline{w}_d))$, with $x_i = g^{w_i}$ and $\overline{x}_i = g^{\overline{w}_i}$ for $i = 1, \ldots, d$. The w_i and $\overline{w}_i$ are chosen at random from $\mathbb{Z}_q$. Note that both these instances use the same pair $(g, \mathcal{G})$. The root of the authentication tree, a_1^1, is computed as $a_1^1 = g^{z_1^1}$, where z_1^1 is chosen at random from $\mathbb{Z}_q$. The initialization phase of

$\Sigma_{\mathcal{P}}$ is completed when the public key of the signer, $(x, \overline{x}, a_1^1)$, is placed in the public directory.

We will now show how the signer computes the first signature on a message $m \in \{0,1\}^{d \cdot l}$, where $m = m_1 || \ldots || m_d$ and the m_i are l-bitstrings, to be interpreted as members of $[0 \ldots 2^l)$.

First, he computes a_0^1 as $a_0^1 = g^{z_0^1}$, with z_0^1 chosen at random from $\mathbb{Z}_q$, and r_0^1 as $r_0^1 = z_0^1 + m_1 w_1 + \cdots + m_d w_d$. Before establishing an authentication for a_0^1, he computes a_1^2 and a_1^3 (in the same way as a_1^1). Next, a_0^1 is authenticated, together with a_1^2 and a_1^3, by computing r_1^1 as $r_1^1 = z_1^1 + \mu_1 \overline{w}_1 \cdots + \mu_d \overline{w}_d$, where $\mu_1 || \cdots || \mu_d = a_1^2 || a_1^3 || a_0^1$. The μ_i are l-bitstrings, to be interpreted as members of $[0, \ldots, 2^l)$.

The values r_0^1, r_1^1, a_0^1, a_1^2 and a_1^3 are forwarded to the receiver, who checks whether

1. $g^{r_0^1} \stackrel{?}{=} a_0^1 \cdot x_1^{m_1} \cdots x_d^{m_d}$, and
2. $g^{r_1^1} \stackrel{?}{=} a_1^1 \cdot \overline{x}_1^{\mu_1} \cdots \overline{x}_d^{\mu_d}$.

Note that the values a_1^2 and a_1^3 are ready to play the role of a_1^1 in the second and third execution of $\Sigma_{\mathcal{P}}$, i.e., to authenticate a_0^2, a_1^4, a_1^5, and a_0^3, a_1^6, a_1^7, respectively. Additionally, however, an authentication *path* has to be given in any execution after the first one, to trace an authenticating node a_1^i back to a_1^1. An authentication path for a node a_1^i consists of all tuples $(a_1^j, a_1^{2j}, a_1^{2j+1}, a_0^j, r_1^j)$ such that a_1^j is an ancestor of a_1^i. For example, an authentication path for a_1^5 would effectively consist of $(a_1^2, a_1^3, a_0^1, r_1^1, a_1^4, a_1^5, a_0^2, r_1^2)$, and the receiver will have to perform the necessary verifications.

We get signatures of length $O(k)$ bits, where k is the number of bits needed to represent an element in $\mathcal{G}$. Moreover, one authentication step requires a constant number of exponentiations in $\mathcal{G}$, both for signing and verification. Note that the 1 exponentiation needed from the signer uses input independent from the bits authenticated $(c_1, \ldots, c_d)$. Therefore we can use the idea suggested by Schnorr of having the signer can precompute this exponentiation if some idle time is available on his computer. This way the on-line time to generate a signature becomes almost negligible.

Previously, the only known way to get a signature scheme provably secure based on discrete log was to use the method from [4] to build a collision intractable hash function and then use Merkle's construction. This would require an exponentiation for each bit processed in the hashing, and moreover we would need as a part of the signature a full preimage under the hash function to authenticate 1 bit. Therefore we would get signatures of length $O(k^2)$ bits and would need $O(k^2)$ exponentiations to make a signature.

7 Conclusion

We have shown that the existence of *signature protocols* is a sufficient condition for the existence of signature schemes that are not existentially forgeable under adaptively chosen message attacks, which is the strongest notion of security for signature schemes (see [9]). The length of the signatures in our schemes grows logarithmically in the number of signatures. In addition to the existence of claw-free pairs of trapdoor one-way permutations, on which the scheme from [9] is based, the most general computational assumption we have been able to find, sufficient for the existence of signature protocols, is the existence of one-way group homomorphisms. As an example, we have presented a signature scheme whose security is equivalent to the difficulty of computing discrete logarithms.

Acknowledgements

It's a pleasure to thank Berry Schoenmakers for many valuable discussions and comments. Also thanks to Matt Franklin for commenting on an earlier version of this paper.

References

[1] M. Abadi, E. Allender, A. Broder, J. Feigenbaum and L. Hemachandra: *On Generating Solved Instances of Computational Problems*, Proc. of Crypto 88, Springer Verlag LNCS series.

[2] J.Bos and D.Chaum: *Provably Unforgeable Signatures*, Proc. of Crypto 92, Springer Verlag LNCS series.

[3] E. F. Brickell and K. S. McCurley: *An Interactive Identification Scheme Based on Discrete Logarithms and Factoring*, Journal of Cryptology, 5(1), pp.29–39, 1992.

[4] I.Damgård: *Collision Free Hash Functions and Public-Key Signature Schemes*, Proc. of EuroCrypt 87, Springer Verlag LNCS series.

[5] C.Dwork, M.Naor: *An Efficient Existentially Unforgeable Signature Scheme and its Applications*, to appear in the Proceedings of Crypto'94, Santa Barbara, August 1994, Springer Verlag LNCS series.

[6] U.Feige, A.Shamir: *Witness Indistinguishable and Witness Hiding Protocols*, Proc. of STOC 90.

[7] U. Feige, A. Fiat and A. Shamir: *Zero-Knowledge Proofs of Identity*, Journal of Cryptology 1 (1988) 77–94.

[8] O. Goldreich: *Two Remarks concerning the GMR Signature Scheme*, Proc. of Crypto 86, Springer Verlag LNCS series.

[9] 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.

[10] L. Guillou and J.-J. Quisquater: *A Practical Zero-Knowledge Protocol fitted to Security Microprocessor Minimizing both Transmission and Memory*, Proc. of EuroCrypt '88, Springer Verlag LNCS series.

[11] R.C.Merkle: *A Digital Signature Based on a Conventional Encryption Function*, Proc. of Crypto 87, Springer Verlag LNCS series.

[12] M.Naor and M.Yung: *Universal One-Way Hash Functions and their Cryptographic Applications*, Proc. of STOC 89.

[13] T. Okamoto: *Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes*, Proc. of Crypto '92, pp.31–53, Santa Barbara, August 1992.

[14] J.Rompel: *One-Way Functions are Necessary and Sufficient for Secure Signatures*, Proc. of STOC 90.

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

[16] M.Tompa and H.Woll: *Random Self-Reducibility and Zero-Knowledge Proof of Information Possession*, Proc. of FOCS 87.

Recent Publications in the BRICS Report Series

RS-94-29 Ivan Damgård and Ronald Cramer. *Secure Signature Schemes Based on Interactive Protocols*. September 1994. 24 pp.

RS-94-28 Oded Goldreich. *Probabilistic Proof Systems*. September 1994. 19 pp.

RS-94-27 Torben Braüner. *A Model of Intuitionistic Affine Logic from Stable Domain Theory (Revised and Expanded Version)*. September 1994. 19 pp. Full version of paper appearing in: ICALP '94, LNCS 820, 1994.

RS-94-26 Søren Riis. *Count(q) versus the Pigeon-Hole Principle*. August 1994. 3 pp.

RS-94-25 Søren Riis. *Bootstrapping the Primitive Recursive Functions by 47 Colors*. August 1994. 5 pp.

RS-94-24 Søren Riis. *A Fractal which violates the Axiom of Determinacy*. August 1994. 3 pp.

RS-94-23 Søren Riis. *Finitisation in Bounded Arithmetic*. August 1994. 31 pp.

RS-94-22 Torben Braüner. *A General Adequacy Result for a Linear Functional Language*. August 1994. 39 pp. Presented at MFPS '94.

RS-94-21 Søren Riis. *Count(q) does not imply Count(p)*. July 1994. 55 pp.

RS-94-20 Peter D. Mosses and Martín Musicante. *An Action Semantics for ML Concurrency Primitives*. July 1994. 21 pp. To appear in Proc. FME '94 (Formal Methods Europe, Symposium on Industrial Benefit of Formal Methods), LNCS, 1994.

RS-94-19 Jens Chr. Godskesen, Kim G. Larsen, and Arne Skou. *Automatic Verification of Real–Timed Systems Using* EPSILON. June 1994. 8 pp. Appears in: Protocols, Specification, Testing and Verification PSTV '94.