New Paradigms for Digital Signatures and Message Authentication Based on Non-Interactive Zero Knowledge Proofs

Mihir Bellare* Shafi Goldwasser†

MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139

Abstract

Using non-interactive zero knowledge proofs we provide a simple new paradigm for digital signing and message authentication secure against adaptive chosen message attack.

For digital signatures we require that the non-interactive zero knowledge proofs be *publicly verifiable*: they should be checkable by anyone rather than directed at a particular verifier. We accordingly show how to implement non-interactive zero knowledge proofs in a network which have the property that anyone in the network can individually check correctness while the proof is zero knowledge to any sufficiently small coalition. This enables us to implement signatures which are history independent.

1 Introduction

1.1 A NIZK Proof Based Paradigm

We show how to use the random functions of [GGM] and the primitive of non-interactive zero-knowledge proof systems introduced by [BFM] to obtain new paradigms for creating digital signatures secure against adaptive chosen message attack. Namely, any method that yields non-interactive zero-knowledge proof-systems, together with the existence of one-way functions, implies new ways to sign digitally. Our digital signature construction is much simpler than all known constructions which

* Supported in part by NSF grant CCR-87-19689 and DARPA Contract N00014-89-J-1988

† Supported in part by NSF grant CCR-86-57527, DARPA Contract N00014-89-J-1988, and by a US Israel binational grant

G. Brassard (Ed.): Advances in Cryptology - CRYPTO '89, LNCS 435, pp. 194-211, 1990.

are secure against adaptive chosen message attack, putting the burden of the work on non-interactive zero-knowledge proof-systems. Moreover, the signatures produced are independent of previous signatures as long as the non-interactive proofs have this property.

Another application of these ideas is to cryptographic protocols. For example, we show how to implement a memoryless system for distributing and checking secret identification numbers, such as phone calling cards, passwords, etc. By memoryless, we mean that no secret data need be stored to *verify* the correctness of id numbers. The only piece of secret information is kept by the center which *generates* the original secret identification numbers.

1.2 Non-Interactive Zero Knowledge Proof Systems

The paradigms we present in this paper are based on the general ability to provide non-interactive zero knowledge proofs.

Informally, a non-interactive zero knowledge proof of a theorem (NP statement) T is a method by which A can give B a string m such that B, upon examining m, is convinced that the theorem is true, but he obtains zero knowledge about the proof.

We will actually be interested in using non-interactive zero knowledge proofs (henceforth abbreviated NIZK proofs) in a public key network. We will want that

- For any pair of users A and B in the network it is the case that A can send NIZK proofs to B
- The number of theorems that can be proved to B is an arbitrary polynomial in the security parameter.

Such a public key network provides the natural setting for digital signatures.

Non-interactive zero knowledge proofs were introduced in [BFM]. The current literature contains a number of implementations of non-interactive zero knowledge proof systems in models that differ in some of their characteristics. In §3 we give a formal definition of NIZK proof systems suitable for our purposes, and consider the models and implementations available.

For digital signatures we need in addition that the proofs be publicly verifiable. This raises some new issues.

1.3 Publicly Verifiable NIZK Proof Systems

Consider a proof of a theorem T provided by A. There are two possibilities with respect to its verifiability:

(1) The proof is directed at a particular person B and only he can verify it

(2) The proof can be verified by any user in the system.

In the latter case we call the proof *publicly verifiable*. The distinction is important for our applications to digital signatures where the public verifiability of proofs will correspond to signatures which anyone can check.

A new security issue that arises with publicly verifiable proofs is maintaining zero knowledge with respect to a coalition of users: although the proof is by definition zero knowledge to each user, can a group of users combine to extract knowledge from it, and, if so, how large should this group be to do damage?

We implement publicly verifiable NIZK proofs addressing these issues in §6.

We emphasize again, however, that the paradigms we use in our applications do not make reference to any particular implementation of NIZK proofs and rely only on their properties as given in §3 together with public verifiability.

1.4 Random Functions

[GGM] introduced the concept of a *pseudo-random collection of functions*. This is a collection of functions $F_k = \{f_s : |s| = k\}$ such that no probabilistic polynomial time algorithm can distinguish a member f_s of F_k from a truly random function. That is, an algorithm whose only access to a function is through queries of its values at various points will be unable to tell whether he is dealing with a member of f_s or with a truly random function.

Through the work of [GGM] and [ILL] we have

Theorem 1.1 The existence of one-way functions implies the existence of pseudo-random collections of functions.

1.5 Related Results

The ideas we present here have already found other applications.

Micali [M2] has observed that by using interactive proofs rather than non-interactive ones in our signature scheme one can implement *undeniable signatures* [CV] based only on the existence of any one-way function.

Feige and Shamir [F] showed that *witness hiding proofs* [FS] suffice to implement a modified version of our signature scheme; we will discuss their result and its relation to ours further in §6.3.

2 Notation

We use [GMR]'s notation and conventions for probabilistic algorithms.

We emphasize the number of inputs received by an algorithm as follows. If algorithm A receives only one input we write "$A(\cdot)$"; if it receives two we write "$A(\cdot,\cdot)$", and so on. If A is a probabilistic algorithm then, for any input i the notation $A(i)$ refers to the probability space which to the string σ assigns the probability that A, on input i, outputs σ (in the special case that A takes no inputs, A refers to the algorithm itself whereas the notation $A()$ refers to the probability space obtained by running A on no inputs). Sometimes we wish to make the coin tosses explicit; in this case we write for example $A(\sigma; i)$ for the output of A on input i with coins σ.

If S is a probability space we denote by $[S]$ the set of elements to which S assigns positive probability.

If $f(\cdot)$ and $g(\cdot,\cdots)$ are probabilistic algorithms then $f(g(\cdot,\cdots))$ is the probabilistic algorithm obtained by composing f and g (i.e. running f on g's output). For any inputs $x, y, \ldots$ the associated probability space is denoted $f(g(x, y, \cdots))$.

If S is a probability space then $x \leftarrow S$ denotes the algorithm which assigns to x an element randomly selected according to S. If S is a finite set, this denotes the operation of selecting an element of S at random.

For probability spaces $S, T, \ldots$, the notation

$$\boldsymbol{P}(p(x, y, \cdots) : x \leftarrow S; y \leftarrow T; \cdots)$$

denotes the probability that the predicate $p(x, y, \cdots)$ is true after the (ordered) execution of the algorithms $x \leftarrow S$, $y \leftarrow T$, etc.

A PPT algorithm means a probabilistic polynomial time algorithm. We assume that a natural encoding of algorithms as binary strings is used.

If A and B are interactive TMs then $(A \leftrightarrow B)(x)$ denotes the probability space of their outputs (some of these outputs might be common while others are private to one or the other of the parties) on input x. As for algorithms, we write for example $B(\sigma)$ for the interactive TM B running with coins σ.

3 Non-Interactive Zero Knowledge Proof Systems

3.1 Definition

A non-interactive zero knowledge proof system for a language L consists of two stages. The first stage, which could be interactive, establishes some information common to the prover and the verifier as well as (possibly) some private information for each. This pre-processing is performed independently of the theorems to be proved. In a second stage the prover chooses and proves theorems to the verifier in zero knowledge, based on the information from the first stage. This theorem proving stage is non-interactive.

The prover and verifier are regarded accordingly as pairs $P = (P_1, P_2)$ and $V = (V_1, V_2)$. The separation between the stages is total. For example, P_1 and P_2 might be completely distinct: the former a trusted center and the latter a prover in the usual sense who sees only information that the center gives him.

Let k be a security parameter.

Definition 3.1 Suppose P_1, P_2, V_1 are probabilistic interactive TMs and V_2 is a deterministic interactive TM, all polynomial time. The pair $P = (P_1, P_2), V = (V_1, V_2)$ constitute a *non-interactive proof system* for a language L if for all sufficiently large k the requirements of the following two stages are met :

(1) **Pre-Processing Stage:**

P_1 interacting with V_1 yields three outputs p, S_P, S_V of which p is common to P_1 and V_1 while S_P is private to P_1 and S_V is private to V_1. If P_1 does not obey the protocol then with high probability V_1 outputs "cheating".

(2) **Theorem Proving Stage:**

P_2 proves theorems to V_2 without interaction (the communication on any input is a single message from P_2 to V_2).

- *Completeness:* For every $x \in L \cap \{0,1\}^k$

$$\boldsymbol{P}(\, V_2(1^k, p, S_V, x, \beta) = \text{ accept } : (p, S_P, S_V) \leftarrow (P_1 \leftrightarrow V_1)(1^k)\,; \\ \beta \leftarrow P_2(1^k, p, S_P, x)\,) \geq 1 - 2^{-2k}\,.$$

- *Soundness:* For every interactive TM $\hat{P}_2$ and every $x \in \overline{L} \cap \{0,1\}^k$,

$$\boldsymbol{P}(\, V_2(1^k, p, S_V, x, \beta) = \text{ accept } : (p, S_P, S_V) \leftarrow (P_1 \leftrightarrow V_1)(1^k)\,; \\ \beta \leftarrow \hat{P}_2(1^k, p, S_P, x)\,) \leq 2^{-2k}\,.$$

In some cases we will also allow the proof system to have as additional input à history of previous theorems and their proofs.

In order to define zero knowledge we first need the concept of a distinguisher.

Definition 3.2 A *distinguisher* for a language L is a pair (I, ρ) where ρ is a deterministic polynomial time predicate and I is a PPT interactive TM such that $I(1^k, \cdot, \cdot, \cdot) \in L \cap \{0,1\}^k$ for all k.

Notation: For any interactive TM $\hat{V}_1$ we let $(P_1.\hat{V}_1)(1^k)$ denote the probability space consisting of the outputs of the interaction together with the history of the interaction and the coin tosses of $\hat{V}_1$. Coupling the last two into a single entity h we write (p, S_P, S_V, h) for an element of this space.

Notation: Let $P = (P_1, P_2), V = (V_1, V_2)$ be a non-interactive proof system for L. For any PPT interactive TM $\hat{V}_1$ and distinguisher (I, ρ) we let

$$p_k(P, \hat{V}_1, (I, \rho)) \;=\; \boldsymbol{P}(\, \rho(1^k, \sigma, p, S_V, h, x_1\beta_1 \ldots x_{k^d}\beta_{k^d}) = 1 : \\ (p, S_P, S_V, h) \leftarrow (P_1.\hat{V}_1)(1^k)\,;\; \sigma \leftarrow \{0,1\}^{k^e}\,; \\ x_1 \leftarrow I(\sigma; 1^k, p, S_V, h, \epsilon)\,;\; \beta_1 \leftarrow P_2(1^k, p, S_P, x_1)\,;\; \ldots\,; \\ x_{k^d} \leftarrow I(\sigma; 1^k, p, S_V, h, x_1\beta_1 \ldots x_{k^d-1}\beta_{k^d-1})\,;\; \beta_{k^d} \leftarrow P_2(1^k, p, S_P, x_{k^d})\,)\,,$$

where $d, e > 0$ are constants determined by the running time of I.

Definition 3.3 A pair of sequences $\{p_k\}, \{q_k\}$ of probabilities are *indistinguishable* (written $\{p_k\} \cong \{q_k\}$) if for all $c > 0$ and sufficiently large k it is the case that $|p_k - q_k| < k^{-c}$.

Definition 3.4 A non-interactive proof system $P = (P_1, P_2), V = (V_1, V_2)$ for L is said to be a non-interactive zero knowledge proof system if there is a a pair $M = (M_1, M_2)$ of PPT algorithms such that the following is true: for any PPT interactive TM $\widehat{V}_1$ and any distinguisher (I, ρ) it is the case that $\{p_k(P, \widehat{V}_1, (I, \rho))\} \cong \{q_k(M_1^{\widehat{V}_1}, M_2, (I, \rho))\}$, where

$$\begin{aligned} q_k(M_1^{\widehat{V}_1}, M_2, (I, \rho)) \;=\; & \boldsymbol{P}(\, \rho(1^k, \sigma, p, S_V^*, h^*, x_1\beta_1^* \ldots x_{k^d}\beta_{k^d}^*) = 1 : \\ & (p^*, S_V^*, h^*) \leftarrow M_1^{\widehat{V}_1}(1^k)\,;\; \sigma \leftarrow \{0,1\}^{k^c}\,; \\ & x_1 \leftarrow I(\sigma; 1^k, p^*, S_V^*, h^*, \epsilon)\,;\; \beta_1^* \leftarrow M_2(1^k, p^*, S_V^*, x_1)\,;\; \ldots\,; \\ & x_{k^d} \leftarrow I(\sigma; 1^k, p^*, S_V^*, h^*, x_1\beta_1^* \ldots x_{k^d-1}\beta_{k^d-1}^*)\,; \\ & \beta_{k^d}^* \leftarrow M_2(1^k, p^*, S_V^*, x_{k^d})\,)\,. \end{aligned}$$

($M_1^{\widehat{V}_1}$ means M_1 with $\widehat{V}_1$ as an oracle). We call $M = (M_1, M_2)$ the *simulator*.

Definition 3.5 A language L is said to have a non-interactive zero knowledge proof (NIZK proof) if there exist $P = (P_1, P_2), V = (V_1, V_2)$ which constitute a non-interactive zero knowledge proof system for it.

3.2 Remarks

General:

We assume that V_2 is deterministic only for simplicity and because it is the case in all current implementations. For greater generality we could allow it coins. Notice that although this would enhance its power in the theorem proving stage the definition of zero knowledge would not have to change: the non-interaction implies that the view of the verifier need not include the coin tosses of V_2.

The Pre-Processing Stage:

Our definition is more general than we need. In the application to signatures there is implicitly a trusted center: since the signer will be proving theorems it is to his advantage to have correct information from the pre-processing stage. In general, however, we must guarantee the privacy and make sure that neither party can cheat.

The Theorem Proving Stage:

The reason for the extremely small error probability of 2^{-2k} in the completeness and soundness conditions is as follows. Since the prover can pick his theorem after seeing the output of the first stage, we would like that for most such outputs, there does not *exist* a false statement that can be proved. Our condition guarantees this. More precisely, if we call $(p, S_P, S_V) \in (P_1 \leftrightarrow V_1)(1^k)$ *good* if $V_2(S_V)$ accepts $P_2(S_P)$'s proof for all $x \in L \cap \{0,1\}^k$ and does not accept any proof for all $x \in \overline{L} \cap \{0,1\}^k$ then

$$\boldsymbol{P}(\, (p, S_P, S_V) \text{ is good } : (p, S_P, S_V) \leftarrow (P_1 \leftrightarrow V_1)(1^k)\,) \geq 1 - 2^{-k}\,.$$

The Zero Knowledge:

A distinguisher (I, ρ) models a two stage process. The first is a requesting stage in which I, given (p, S_V, h) from the pre-processing stage, asks the prover to supply zero knowledge proofs, with respect to p, of theorems of its choice. Its requests are

adaptive, depending on responses to previous requests. This corresponds to what we will want with digital signatures where the theorems that the signer proves are chosen by an adversary who has seen the public key and responses to previous requests for signatures. In a second stage the predicate ρ is computed on the received information. Our definition of zero knowledge asks that there be a simulator which could interact with this process I in such a way that it would not know that it was not interacting with the real prover; that is, the value of the predicate ρ would be 1 with essentially the same probability in both cases.

Our simulator accordingly simulates the two stages independently. It first creates information (p^*, S_V^*, h^*) pertaining to the pre-processing stage. This is done by M_1 who has blackbox access to a possibly cheating verifier $\hat{V}_1$. In the second stage M_2 is used to supply simulations of zero knowledge proofs that I requests.

A more general definition would be that for every $\hat{V}_1$ there was such an M_1, but to avoid a profusion of quantifiers we stick to the simple case.

Public Verifiability:

The definition as it stands does not address public verifiability. For publicly verifiable proofs the secret information of the the verifier is either void, or there are lots of verifiers each of whom has his own secret. This is considered further in §6.

3.3 A Look at Available Implementations

All the models described in the literature so far do fit our definition. We discuss briefly here these models and implementations.

The first model of NIZK proofs was introduced by Blum, Feldman and Micali [BFM][1]. In this model the prover and the verifier share a common random string with respect to which the prover provides his proofs. In terms of our definition, p is the common random string and there are no private inputs from the first stage (S_P and S_V are both the empty string).

The common string must be random to ensure both the validity of the proofs and their zero-knowledge. For our application to digital signatures we would have the legal signer publish a random string in his public file and give proofs with respect to it. Other users, functioning as verifiers, would read this and check his proofs. Since the signer wishes to avoid forgeries, it is to his interest to pick the string to be truly random, and he will do so. Note that the proofs are indeed publicly verifiable. The drawback of the current implementation however [M1], is that a proof depends on previous proofs resulting in signatures which depend on previous ones.

The model of Kilian, Micali and Ostrovsky [KMO] has an initial interactive pre-processing stage between prover and verifier. This model fits our definition with p being the empty string, S_P the pair of seeds (s_0, s_1) that P oblivious transfers to

[1] The implementation described in the original [BFM] paper is not known to have a proof; a correct scheme has been announced by S. Micali [M1]. However we do not know whether this scheme satisfies our more stringent zero knowledge requirements. It does however satisfy the more relaxed requirement stated in Appendix B which, as we indicate there, is really all we need for signatures.

V, and S_V the seed (either s_0 or s_1) that V receives. However the fact that there is an interaction between prover and verifier means that we cannot use it for the applications described in this paper.

Proposed in [BM2] is a public key system under which Kilian-Micali-Ostrovsky type oblivious transfer based proofs can be implemented. In this model a single verifier can receive proofs from many provers and our message authentication between pairs of users (§5.1) can be implemented in this system. Moreover, some of the implementations in [BM2] are quite efficient. In terms of our definition, p would be the verifier's public key and S_V his secret key while S_P would be the empty string.

4 NIZK Proofs and Digital Signatures

Here we show how NIZK proofs together with one-way functions yield digital signatures.

4.1 How to Sign

Let E be a probabilistic public key encryption algorithm [GM] (we implement this via the bit commitment scheme of Naor [N] which is based on any one-way function). Let $F_k = \{f_s : |s| = k\}$ denote a collection of pseudo random functions as defined in [GGM]. Assume we have a publicly verifiable NIZK proof system for some NP complete language. For our purposes the pre-processing stage of this NIZK proof system is best thought of as an algorithm Z which on input a security parameter outputs just some public information p. We denote by $NIZK_p(T)$ a NIZK proof of T with respect to p.

User U's public file is $PK_U = (1^k, E, \alpha, p)$ and his secret file is $SK_U = (r, s)$, where

- k is the security parameter
- $s \in \{0,1\}^k$ is the randomly chosen index to a pseudo-random function from F_k
- $\alpha = E(r, s)$ is an encryption of s.
- $p \leftarrow Z(1^{k^c})$ is the information needed for the NIZK proofs (c is a constant).

Then, the digital signature of a document D is defined to be

$$\sigma(D) = (D, R, NIZK_p(T_{PK,D,R}))$$

where $R = f_s(D)$ and $T_{PK,D,R}$ is defined to be the NP statement

$$\exists s \exists r \, [\alpha = E(r, s) \text{ and } R = f_s(D)] \ .$$

Since $T_{PK,D,R}$ is an NP statement, there exists a non-interactive zero-knowledge proof $NIZK_p(T_{PK,D,R})$ of it. The public verifiability of the NIZK proof means that this is really a signature: anyone can check the validity of $\sigma(D)$ given the signer's public key PK_U.

4.2 Comparison with Previous Signature Schemes

In all previous signature schemes secure against adaptive chosen message attack [GMR], [BM1], [NY] the signature of the i-th document D_i was a function of all the previous signatures of documents $D_1, \ldots, D_{i-1}$ (we will refer to such schemes as *history dependent*). The sizes of signatures thus quickly become very impractical. Although methods for dealing with this problem and improving the efficiency of these schemes have been suggested (by Levin, with improvements by Goldreich [G]), these methods are complicated and cumbersome. No such problems exist with our scheme where the signature of a document is independent of previous signatures, as long as the underlying NIZK proofs have the property of being independent of previous proofs. Our implementations of NIZK proofs in §6 do have this property.

4.3 Assumptions

In terms of assumptions we have shown that a digital signature scheme secure against adaptive chosen message attack exists if one-way functions and publicly verifiable NIZK proof systems exist. It is known that digital signature schemes secure against adaptive chosen message attack exist if one-way permutations exist [NY], and it is clear that the existence of digital signatures implies the existence of one-way functions. The relation of NIZK proofs to one-way permutations is not known.

4.4 Security

We recall that in an *adaptive chosen message attack* an adversary can request from the true signer the signature of any number of messages of his choice. Moreover, he can ask for these signatures one by one, with his requests depending on the signatures provided in response to previous requests. The scheme is secure against adaptive chosen message attack if, after requesting signatures in this fashion, the adversary remains unable to forge the signature to *any* message whose signature he has not previously seen. This notion of security, which was introduced in [GMR], represents the strongest possible natural notion of security for a digital signature scheme.

Theorem 4.1 The digital signature scheme of §4.1 is secure against adaptive chosen message attack.

A very rough intuition of why this is true is the following. By [GGM] an adversary who can request to see the value of a random function f_s on a polynomial number of strings $D_1, D_2, \ldots$ of his choice cannot compute even one additional pair D, R such that $R = f_s(D)$ and $D \neq D_i$ for all i. The only difference in our scenario is that the adversary sees along with the set of $\{D_i, R_i\}$ where $R_i = f_s(D_i)$ for all i, many $NIZK_p(T_{D_i})$, i.e proofs that in fact R_i is the result of applying f_s to D_i. But, since these are zero-knowledge proofs they convey no extra knowledge and the [GGM] proof applies.

A more complete proof is in Appendix A.

5 Further Applications of the NIZK Paradigm

The NIZK proofs based paradigm of the previous section can also be adapted to message authentication and the memoryless distribution of id numbers.

5.1 Message Authentication between Pairs of Users

Note that the property of public verifiability is not necessary for message authentication between two users. We simply replace the use of publicly verifiable proofs as in §4 by NIZK proofs from A to B.

Efficient implementations of NIZK proofs between pairs of users are known [BM2].

5.2 Memoryless Distribution of Identification Numbers

Consider an application where a central authority like the phone company, or a passport producing facility needs to generate unique unforgeable id numbers for its users. The users should be able to present their identification numbers in numerous distributed local stations, and the local station should have the capability to check the validity of the id.

One previous solution to this problem was presented by [GGM] where they used random functions applied to the user name to create the user id number. The disadvantage of that proposal was that the all the local stations needed to keep secret the index to the random function. Using non-interactive zero-knowledge proofs that disadvantage can now be removed. The idea now is for the center alone to keep secret the single index s to the random function, together with a value r, and for the center to publish in a public file the pair (E, α) where $\alpha = E(r, s)$. When user U needs an id, the center computes $I = f_s(U)$ and gives I to U along with a non-interactive zero-knowledge proof, $NIZK_{center}(T)$ where T is the NP statement

$$\exists s \exists r \, [\alpha = E(r, s) \text{ and } I = f_s(U)] \ .$$

The local center has no knowledge of U or any special information whatsoever. Whenever U needs to authenticate himself, he simply shows the local center I and $NIZK_{center}(T)$ which convinces the local center that the user possesses a legal id number.

6 NIZK Proof Systems with Public Verifiability

We sketch here very briefly some implementations of publicly verifiable NIZK proof systems. In these systems there is a center who publishes some information p that everyone can see, and then gives each user B some secret information S_B. Only the public part p is necessary to prove theorems, and anyone with a secret key can check such a proof.

6.1 A Simple Scheme

We can get a first, simple scheme using methods similar to [BM2]. Let k be the usual security parameter. The center picks at random a number of probabilistic encryption algorithms [GM] with their corresponding decryption keys; let these be

$$\{(E_{i,j}, D_{i,j}) : i = 1, \ldots, k \text{ and } j = 0,1\} .$$

The center now publishes

$$p = ((E_{1,0}, E_{1,1}), (E_{2,0}, E_{2,1}), \ldots, (E_{k,0}, E_{k,1}))$$

as a key visible to all users in the system. To any user B who wishes to be able to verify proofs, the center, having picked $j_1, \ldots, j_k \in \{0,1\}$ at random, sends[2]

$$S_B = (D_{1,j_1}, D_{2,j_2}, \ldots, D_{k,j_k}) .$$

User B makes this his secret key.

The following encoding of proofs was used by Kilian, Micali and Ostrovsky [KMO] (it arises from Blum's ZK proof of Hamiltonian cycle) and will allow us to prove theorems in zero knowledge.

Theorem 6.1 [KMO] Suppose A has a NP theorem T and a proof of it. Then she can compute a sequence

$$s = ((s_{1,0}, s_{1,1}), (s_{2,0}, s_{2,1}), \ldots, (s_{k,0}, s_{k,1}))$$

of pairs of strings which encode a proof of T in the following sense:

(1) if the proof is incorrect then for each $i = 1, \ldots, k$ there is a $j \in \{0,1\}$ such that seeing $s_{i,j}$ will reveal the incorrectness of the proof

(2) if for each $i = 1, \ldots, k$ one does not see both $s_{i,0}$ and $s_{i,1}$ then the proof is zero knowledge.

A's proof of T is now just

$$NIZK_p(T) = ((r_{1,0}, r_{1,1}), (r_{2,0}, r_{2,1}), \ldots, (r_{k,0}, r_{k,1})) ,$$

where $r_{i,j} = E_{i,j}(s_{i,j})$. Each user B can decrypt exactly one of $r_{i,0}$ and $r_{i,1}$ at random for each i. Theorem 6.1 thus guarantees that this is a non-interactive zero knowledge proof system.

In the terminology of [BM2], we have established oblivious transfer channels; however in our case the proof does not depend on any particular verifier but only on p.

Note that in this scheme the proof of a theorem does not depend on the proofs of previous theorems in contrast to [BFM]+[M1].

Also note that A is not special: any user can use the central public key p to provide proofs that all the others can check. All users can thus do digital signatures, publishing in their individual files the necessary information as described in §4 and using the central key for the NIZK proofs.

The drawback of this scheme is that two bad users could combine to break it. If a pair of users put their secret keys together they might know both $D_{i,0}$ and $D_{i,1}$ for

[2] If the center is not trusted, oblivious transfer can be used here instead, as described in [BM2].

some i and then the proof would not be zero knowledge to them. They could then forge signatures. The next scheme is more robust in this regard.

6.2 Zero Knowledge to Many Users Simultaneously

Here we show how to provide security against any $O(k)$ users combining. However, the size of the proofs will grow as a function of k. The key to the stronger publicly verifiable NIZK system is the use of a different method, due to Kilian [K], for encoding proofs.

Theorem 6.2 [K] Suppose that A has a theorem T of size n and a proof of it. Then she can construct a *tableau* $\mathcal{T}(T)$ for T. This consists of a sequence $(\mathcal{T}_1, \ldots, \mathcal{T}_{nkp(nk)})$, where each $\mathcal{T}_i$ is a sequence of $p(nk)$ strings (p is some fixed polynomial), such that the following are true:

(1) If $\mathcal{T}(T)$ does not encode a correct proof of T then there is a "check" that reveals this. A check is some predicate that is evaluated on some four positions in the sequence $\mathcal{T}_i$ and there is a total of $p(nk)$ such checks.

(2) If for all i one sees $\leq k-1$ positions of $\mathcal{T}_i$ then no knowledge about the proof will be revealed.

The strings which constitute the elements of the tableau are of constant length.

For our scheme the center selects at random $nkp(nk)^2$ probabilistic encryption algorithms together with their decryption keys; let these be

$$\{(E_{i,j}, D_{i,j}) : i = 1, \ldots, nkp(nk) \text{ and } j = 1, \ldots, p(nk)\} .$$

The center now publishes

$$p = ((E_{1,1}, \ldots, E_{1,p(nk)}), (E_{2,1}, \ldots, E_{2,p(nk)}), \ldots, (E_{nkp(nk),1}, \ldots, E_{nkp(nk),p(nk)})) .$$

When user B requests a secret key the center picks, for each $i = 1, \ldots, nkp(nk)$, one of the $p(nk)$ checks at random and sends to B

$$S_B = \left\{(D_{i,j_{i,1}}, D_{i,j_{i,2}}, D_{i,j_{i,3}}, D_{i,j_{i,4}}) : i = 1, \ldots, nkp(nk)\right\}$$

where $1 \leq j_{i,1}, j_{i,2}, j_{i,3}, j_{i,4} \leq p(nk)$ are the four tableau positions which make up the chosen check ($i = 1, \ldots, nkp(nk)$). User B makes this his secret key.

When user A wishes to prove a theorem T she makes a tableau $\mathcal{T}(T) = \mathcal{T}_1, \ldots, \mathcal{T}_{nkp(nk)}$ and encrypts each position with the corresponding encryption algorithm from p. Any user B can see exactly one random check per tableau and hence will detect a false proof with probability

$$\geq 1 - \left(1 - \frac{1}{p(nk)}\right)^{nkp(nk)} \geq 1 - e^{-nk} ,$$

by property 1 of a tableau as given in Theorem 6.2.

Any combination of $\leq \frac{k-1}{4}$ users in the system, pooling their secret keys, see $\leq k-1$ positions of each $\mathcal{T}_i$. So by property 2 of Theorem 6.2 the proof will be zero knowledge to this coalition.

6.3 History Independent Signatures

Plugging the above implementation into the signature scheme of §4 yields a history independent signature scheme in this model where there is a network of users each possessing certain special public and secret keys enabling NIZK proofs. The question of whether history independent signatures can be achieved without such pre-processing still remains. Feige and Shamir [F] answered this in the affirmative. They modify our scheme so that it uses witness hiding proofs, and then use the [DMP] implementations of one-theorem NIZK to implement the latter. The resulting scheme is based on the specific assumption of quadratic residuosity.

A Appendix: Proof of Security for the Signature Scheme

Here we give a more formal proof of Theorem 4.1.

Suppose $\mathcal{F}$ is a forger who, after an adaptive chosen message attack, succeeds in forgery with probability $\epsilon(k) \geq k^{-c}$, where $c > 0$ is some constant and the probability is over the choice of the public and secret keys and the coin tosses of both the signer and $\mathcal{F}$. We propose to derive a contradiction.

We let $\mathcal{V}$ denote the verification algorithm: $\mathcal{V}$ takes as input a public key $PK = (1^k, E, \alpha, p)$ and a signature (D, R, β) and outputs 1 iff β is a proof of $T_{PK,D,R}$ with respect to p.

The proof below is for a model, like [BFM], in which the verifier has no secret information (a little more care is required in the case of a model like that of §6 where each verifier has a separate secret, and we discuss this in the final paper). For our purposes the pre-processing stage is replaced by the algorithm Z which on input a security parameter outputs simply some public information p for proofs.

We will consider five experiments. Each experiment has the following format:

- *Make Public Key:* Create some value PK which will serve as the public key
- *Sign:* Invoke the forger $\mathcal{F}$ on PK and respond in some manner to his requests
- *Output:* Output either 0 or 1 based on the success of $\mathcal{F}$.

The experiments are as follows.

$E_0(1^k)$: (True signing process)

- *Make Public Key:* Let $s, r \leftarrow \{0,1\}^k$; $\alpha \leftarrow E(r, s)$; $p \leftarrow Z(1^{k^c})$. Let $PK = (1^k, E, \alpha, p)$.
- *Sign:* Invoke the forger $\mathcal{F}$ on input PK. When he requests the signature of a document D give him the signature $(D, f_s(D), NIZK_p(T_{PK,D,f_s(D)}))$.
- *Output:* Output 1 iff the forger outputs a forgery $(\widehat{D}, \hat{R}, \hat{\beta})$ such that $\mathcal{V}(PK, (\widehat{D}, \hat{R}, \hat{\beta})) = 1$.

$E_1(1^k)$: (Change acceptance criterion)

- *Make Public Key:* Let $s, r \leftarrow \{0,1\}^k$; $\alpha \leftarrow E(r, s)$; $p \leftarrow Z(1^{k^c})$. Let $PK = (1^k, E, \alpha, p)$.
- *Sign:* Invoke the forger $\mathcal{F}$ on input PK. When he requests the signature of a document D give him the signature $(D, f_s(D), NIZK_p(T_{PK,D,f_s(D)}))$.
- *Output:* Output 1 iff the forger outputs a forgery $(\widehat{D}, \hat{R}, \hat{\beta})$ such that $\mathcal{V}(PK, (\widehat{D}, \hat{R}, \hat{\beta})) = 1$ and, in addition, $f_s(\widehat{D}) = \hat{R}$.

$E_2(1^k)$: (Use Simulator for Proofs)

- *Make Public Key:* Let $s, r \leftarrow \{0,1\}^k$; $\alpha \leftarrow E(r,s)$; $p^* \leftarrow M_1(1^{k^c})$[3]. Let $PK = (1^k, E, \alpha, p^*)$.
- *Sign:* Invoke the forger $\mathcal{F}$ on input PK. When he requests the signature of a document D give him $(D, f_s(D), \beta^*)$ where $\beta^* = M_2(1^{k^c}, p^*, T_{PK,D,f_s(D)})$.
- *Output:* Output 1 iff the forger outputs a forgery $(\widehat{D}, \hat{R}, \hat{\beta})$ such that $\mathcal{V}(PK, (\widehat{D}, \hat{R}, \hat{\beta})) = 1$ and, in addition, $f_s(\widehat{D}) = \hat{R}$.

$E_3(1^k)$: **(Change the encryption)**

- *Make Public Key:* Let $s, s', r \leftarrow \{0,1\}^k$; $\alpha \leftarrow E(r,s')$; $p^* \leftarrow M_1(1^{k^c})$. Let $PK = (1^k, E, \alpha, p^*)$.
- *Sign:* Invoke the forger $\mathcal{F}$ on input PK. When he requests the signature of a document D give him $(D, f_s(D), \beta^*)$ where $\beta^* = M_2(1^{k^c}, p^*, T_{PK,D,f_s(D)})$ is the result of running the simulator on input the (false) statement $T_{PK,D,f_s(D)}$.
- *Output:* Output 1 iff the forger outputs a forgery $(\widehat{D}, \hat{R}, \hat{\beta})$ such that $\mathcal{V}(PK, (\widehat{D}, \hat{R}, \hat{\beta})) = 1$ and, in addition, $f_s(\widehat{D}) = \hat{R}$.

$E_4(1^k)$: **(Use a random function)**

- *Make Public Key:* Let $s', r \leftarrow \{0,1\}^k$; $\alpha \leftarrow E(r,s')$; $p^* \leftarrow M_1(1^{k^c})$. Let $PK = (1^k, E, \alpha, p^*)$.
- *Sign:* Invoke the forger $\mathcal{F}$ on input PK. When he requests the signature of a document D give him (D, R, β^*) where $R \leftarrow \{0,1\}^k$ and $\beta^* = M_2(1^{k^c}, p^*, T_{PK,D,R})$ is the result of running the simulator on input the (false) statement $T_{PK,D,R}$.
- *Output:* Let $R' \leftarrow \{0,1\}^k$. Output 1 iff the forger outputs a forgery $(\widehat{D}, \hat{R}, \hat{\beta})$ such that $\mathcal{V}(PK, (\widehat{D}, \hat{R}, \hat{\beta})) = 1$ and, in addition, $\hat{R} = R'$.

For each i let $p_i(k) = \boldsymbol{P}[E_i(1^k) = 1]$.

Fact 1: $p_0(k)$ is by definition the probability $\epsilon(k)$ of successful forgery.

Fact 2: $p_4(k) \leq 2^{-k}$ since the probability that $\hat{R} = R' = 2^{-k}$.

Now

$$\begin{aligned} \frac{\epsilon(k)}{2} &\leq \epsilon(k) - p_4(k) \\ &= p_0(k) - p_4(k) \\ &= \sum_{i=0}^{3} p_i(k) - p_{i+1}(k) \ , \end{aligned}$$

We thus have four cases in each of which we derive a contradiction.

Case 1: $p_0(k) - p_1(k) \geq \frac{\epsilon(k)}{8}$

[3] With the pre-processing stage represented by Z we do not have a history and coin tosses h in the simulator's output.

This is not possible simply by virtue of having a proof system. With probability $\geq 1-2^{-k}$ it is the case that $f_s(\widehat{D}) = \hat{R}$ iff $\mathcal{V}(PK, (\widehat{D}, \hat{R}, \hat{\beta})) = 1$.

Case 2: $p_1(k) - p_2(k) \geq \frac{\epsilon(k)}{8}$.

This would imply an ability to distinguish simulated proofs from real ones. In the final paper we will show how to use the forger to construct a distinguisher (I, ρ) which tells apart real proofs of theorems from simulations of proofs of these theorems.

Case 3: $p_2(k) - p_3(k) \geq \frac{\epsilon(k)}{8}$.

We will distinguish between encryptions of different values.

On input $(1^k, s_0, s_1, \alpha)$ where

$$s_0, s_1 \leftarrow \{0,1\}^k \; ; \; b \leftarrow \{0,1\} \; ; \; \alpha \leftarrow E(s_b) \; ,$$

the following algorithm A predicts b:

- *Make Public Key:* Let $p^* \leftarrow M_1(1^{k^c})$ and set $PK = (1^k, E, \alpha, p^*)$.
- *Sign:* Invoke the forger $\mathcal{F}$ on input PK. When he requests the signature of a document D give him $(D, f_{s_1}(D), \beta^*)$ where $\beta^* = M_2(1^{k^c}, p^*, T_{PK,D,f_{s_1}(D)})$.
- *Output:* Output 1 iff the forger outputs a forgery $(\widehat{D}, \hat{R}, \hat{\beta})$ such that $\mathcal{V}(PK, (\widehat{D}, \hat{R}, \hat{\beta})) = 1$ and, in addition, $f_{s_1}(\widehat{D}) = \hat{R}$.

The probability that A correctly predicts b is

$$\begin{aligned} \boldsymbol{P}[A = b] &= \frac{1}{2}\boldsymbol{P}[A = 1 | b = 1] + \frac{1}{2}\boldsymbol{P}[A = 0 | b = 0] \\ &= \frac{1}{2}p_2(k) + \frac{1}{2}(1 - p_3(k)) \\ &\geq \frac{1}{2} + \frac{\epsilon(k)}{16} \; , \end{aligned}$$

contradicting the indistinguishability of encryptions.

Case 4: $p_3(k) - p_4(k) \geq \frac{\epsilon(k)}{8}$.

We will distinguish [GGM] functions from random functions.

Given an oracle O_f for a function $f : \{0,1\}^k \rightarrow \{0,1\}^k$, the experiment $A^{O_f}(1^k)$ is

- *Make Public Key:* Let $s', r \leftarrow \{0,1\}^k \; ; \; \alpha \leftarrow E(r, s') \; ; \; p^* \leftarrow M_1(1^{k^c})$. Let $PK = (1^k, E, \alpha, p^*)$.
- *Sign:* Invoke the forger $\mathcal{F}$ on input PK. When he requests the signature of a document D give him (D, R, β^*) where $R = O_f(D)$ and $\beta^* = M_2(1^{k^c}, p^*, T_{PK,D,R})$.
- *Output:* Output 1 iff the forger outputs a forgery $(\widehat{D}, \hat{R}, \hat{\beta})$ such that $\mathcal{V}(PK, (\widehat{D}, \hat{R}, \hat{\beta})) = 1$ and, in addition, $\hat{R} = O_f(\widehat{D})$.

Then

$$\begin{aligned} \boldsymbol{P}[A^{O_f} = 1 | f \leftarrow F_k] &= p_3(k) \\ \boldsymbol{P}[A^{O_f} = 1 | f \leftarrow H_k] &= p_4(k) \; , \end{aligned}$$

where H_k denotes the set of all functions from $\{0,1\}^k$ to $\{0,1\}^k$. We thus contradict [GGM].

B Appendix: Using a Simpler Zero Knowledge Definition

Is it possible to prove our signature scheme correct with a definition of zero knowledge which is simpler and more like the usual one? We do not know how to do quite that, but we can show something essentially as good: a modified scheme can be proved correct with a simple definition of NIZK.

A simpler definition of the zero knowledge would be of the following form.

Notation: For any constant $c > 0$ let $L_{k^c} = \underbrace{(L \cap \{0,1\}^k) \times \cdots \times (L \cap \{0,1\}^k)}_{k^c}$.

Definition B.1 Let $P = (P_1, P_2), V = (V_1, V_2)$ be a non-interactive proof system for a language L, and let $\hat{V}_1$ be a PPT interactive TM. Let $c > 0$ be a constant. The *view* of the cheating verifier $\hat{V} = (\hat{V}_1, V_2)$ on any input $\vec{x} \in L_{k^c}$ is

$$\begin{aligned} View_{\hat{V}}(1^k, \vec{x}) \;=\; & \{\, (p, S_V, h, \vec{x}, \vec{\beta}) : (p, S_P, S_V, h) \leftarrow (P_1.\hat{V}_1)(1^k)\,; \\ & \beta_1 \leftarrow P_2(1^k, S_P, p, x_1)\,;\, \ldots\,;\, \beta_{k^c} \leftarrow P_2(1^k, S_P, p, x_{k^c})\,\} \end{aligned}$$

where $\vec{\beta} = (\beta_1, \ldots, \beta_{k^c})$.

Definition B.2 A non-interactive proof system $P = (P_1, P_2), V = (V_1, V_2)$ for L is said to be a non-interactive zero knowledge proof system if for any PPT interactive TM $\hat{V}_1$ and any constant $c > 0$ there is a PPT algorithm M such that the ensembles $\{M(1^k, \vec{x})\}_{\vec{x} \in L_{k^c}}$ and $\{View_{\hat{V}}(1^k, \vec{x})\}_{\vec{x} \in L_{k^c}}$ are computationally indistinguishable, where $\hat{V} = (\hat{V}_1, V_2)$.

(The indistinguishability is as usual in terms of poly-sized families of circuits).

Crucial to constructing a signature scheme secure against adaptive chosen message attack based on this definition is a theorem of Even, Goldreich and Micali [EGM] which states that any scheme secure against random message attack can be transformed into one secure against adaptive chosen message attack. Moreover, a signature in the transformed scheme is independent of previous signatures if this was true for the original scheme. We are thus done given

Theorem B.1 The digital signature scheme of §4 is secure against random message attack under the above definition of NIZK proofs.

Proof: In the final paper. □

References

[BM1] Bellare, M., and S. Micali, "How to Sign Given Any Trapdoor Function," STOC 88.

[BM2] Bellare, M., and S. Micali, "Non-Interactive Oblivious Transfer and Applications," CRYPTO 89.

[BFM] Blum, M., P. Feldman and S. Micali, "Non-Interactive Zero Knowledge and its Applications," STOC 88.

[CV] Chaum, D. and H. Van Antwerpen, "Undeniable Signatures," CRYPTO 89.

[DMP] De Santis, A., G. Persiano and S. Micali, "Non-Interactive Zero Knowledge Proof Systems," CRYPTO 87.

[EGM] Even, S., O. Goldreich and S. Micali, "On-line/Off-line Digital Signatures," CRYPTO 89.

[F] Feige, U., personal communication, September 1989.

[FS] Feige, U. and A. Shamir, "Zero Knowledge Proofs of Knowledge in two Rounds," CRYPTO 89.

[G] Goldreich, O., "Two Remarks Concerning the GMR Signature Scheme," MIT Laboratory for Computer Science Technical Report 715, (September 1986).

[GM] Goldwasser, S., and S. Micali, "Probabalistic Encryption," *Journal of Computer and System Sciences* 28 (April 1984), 270-299.

[GGM] Goldreich, O., S. Goldwasser, and S. Micali, "How To Construct Random Functions," *Journal of the Association for Computing Machinery*, Vol. 33, No. 4 (October 1986), 792-807.

[GMR] Goldwasser, S., S. Micali and R. Rivest, "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks," *SIAM Journal on Computing*, vol. 17, No. 2, (April 1988), 281-308.

[ILL] Impagliazzo, R., L. Levin, and M. Luby, "Pseudo-Random Generation from One-Way Functions," STOC 89.

[K] Kilian, J., "Founding Cryptography on Oblivious Transfer," STOC 88.

[KMO] Kilian, J., S. Micali and R. Ostrovsky, "Efficient Zero Knowledge Proofs with Bounded Interaction," CRYPTO 89.

[M1] Micali, S., personal communication, April 1989.

[M2] Micali, S., personal communication, August 1989.

[N] Naor, M., "Bit Committment using Pseudo-Randomness," CRYPTO 89.

[NY] Naor, M., and M. Yung, "Universal One-Way Hash Functions and their Cryptographic Applications," STOC 89.