Security of Blind Digital Signatures

(Extended Abstract)

Ari Juels[1]* Michael Luby[2] Rafail Ostrovsky[3]

[1] RSA Laboratories. Email: ari@rsa.com.
[2] Digital Equipment Corporation, 130 Lytton Avenue, Palo Alto, CA 94301-1044. Email: luby@pa.dec.com.
[3] Bell Communications Research, 445 South St., MCC 1C-365B, Morristown, NJ 07960-6438, USA. Email: rafail@bellcore.com.

Abstract. Blind digital signatures were introduced by Chaum. In this paper, we show how *security* and *blindness* properties for blind digital signatures, can be simultaneously defined and satisfied, assuming an arbitrary one-way trapdoor permutation family. Thus, this paper presents the first complexity-based proof of security for blind signatures.

1 Introduction

A digital signature scheme allows one to "sign" documents in such a way that everyone can verify the validity of authentic signatures, but no one can forge signatures of new documents. The strongest definition of security for a digital signature scheme was put forth by Goldwasser, Micali, and Rivest [15]. Several schemes, based on both specific and general complexity assumptions, were subsequently shown to satisfy this strongest definition. A variation on basic digital signatures, known as *blind digital signatures*, was proposed by Chaum. Blind digital signature schemes include the additional requirement that a signer can "sign" a document (which is given to him in some "encrypted" form) without knowing what the document contains. Blind digital signatures play a central role in anonymous electronic cash applications. In this paper, we show how security and blindness properties in digital signatures can be simultaneously defined and satisfied, assuming an arbitrary one-way trapdoor permutation family.

While our construction achieves the strongest guarantees under general complexity assumptions and runs in polynomial time (in all the parameters), it is quite complicated and inefficient. The contribution of this paper is therefore twofold: (1) we show that the notions of blindness and security can be simultaneously formalized and (2) we exhibit a "constructive proof of existence" of

* Part of this work was done while this author was at U.C. Berkeley under NSF Grant CCR-9505448.

a blind digital signature scheme which satisfies these strong requirements. The current paper leaves open the question of an efficient implementation. We stress, though, that it was previously not clear whether the strong security guarantees for blind digital signatures could be satisfied under *any* complexity assumptions.

We preface definitions and our main result with some background.

Digital Signatures: Informally, a signature scheme allows a user with a public key and a corresponding private key to sign a document in such a way that everyone can verify the signature of the document (using her public key) but no one else can forge the signature of another document. Digital signatures were originally defined by Diffie and Hellman [9], and the first implementation was based on the RSA trapdoor function [23]. Goldwasser, Micali, and Rivest [15] defined the strongest known "existential adaptive chosen-message attack" against digital signature schemes. They also demonstrated the first scheme which is secure against such an attack[4] assuming the existence of claw-free permutations, which in turn may be based on the hardness of factoring. Subsequently, signatures secure against existential adaptive chosen-message attacks were shown assuming the existence of trapdoor permutations [2], one-way permutations [19], and general one-way functions [24]. More efficient schemes secure against such an attack were shown in [8], and schemes with additional properties were considered in [11, 3, 16].

Blind Signatures: Chaum [6] proposed the notion of "blind digital signatures" as a key tool for constructing various anonymous electronic cash instruments. These are instruments for which the bank cannot trace where (and hence for what purpose) a user spends her electronic currency. In this paper we do not address the broad issues of electronic commerce, but concentrate our attention solely on blind signatures. Informally, a blind digital signature scheme may be thought of as an abstract game between a "user" and a "bank". The user has a secret document for which she needs to get the signature from the bank. She should be able to obtain this signature without revealing to the bank anything about her document except its length. On the other hand, the security of the signature scheme should guarantee that it is difficult for the user to forge a signature of any additional document, even after getting from the bank a number of blind signatures. Blind/untraceable signatures have attracted considerable attention in the literature (see, for example, [7, 20, 1, 22] and references therein), and are used in several proposed electronic digital cash systems. Researchers use two

[4] We remark that [23] is not secure against existential adaptive chosen-message attacks since there are signatures that can be forged under this attack.

different approaches for proving the security of signature schemes: complexity-based proofs of security [9, 15, 2, 19, 24, 3, 16, 8] and random-oracle model proofs of security [10, 4, 21, 22]. Let us elaborate on these two notions of security:

Two Notions of Security for Digital Signatures:

- **Complexity-based proofs:** The complexity-based approach was put forth by Diffie and Hellman [9]. They suggested that the security of a cryptographic primitive could be *reduced* to a hardness assumptions of certain fundamental problems, such as the existence of one-way functions. The approach proved very successful, as a large number of cryptographic primitives, including pseudo-random generators, signatures and secure protocols were shown to exist based on general complexity assumptions.

- **Proofs based on random oracle model:** In the case when complexity-based proofs seem to be difficult to attain, the approach used, for example in [10, 4, 21, 22], is to assume that a cryptographic primitive (such as DES or MD5) behaves like a truly random function. The security of the scheme is then shown under the assumption that the underlying primitive behaves in a near ideal fashion. Such proofs are weaker than complexity-based proofs. (For a related discussion see [5]).

Clearly, the complexity-based proofs of security are preferable to random-oracle model proofs of security. Until now, however, the only proofs of security for blind digital signature schemes have been in the random oracle model. This paper presents the first blind signature scheme with complexity-based proof of security.

Pointcheval and Stern [22] address the security of several blind digital signatures schemes, including blind variants of the Okamoto [20], Schnorr [25], and Guillou-Quisquater [17] signature schemes. In particular, [22] proves the security of Okamoto-Schnorr and Okamoto-Guillou-Quisquater blind signatures in the random oracle model. Thus, Pointcheval and Stern consider blind signatures which rely on number-theoretic assumptions and show proofs of security only in the random-oracle model. In addition, their security proofs, while polynomial in the size of the cryptographic keys, are exponential in the number of blind digital signatures obtained before the break (i.e. if the number of signatures that are required before the break is greater than logarithmic, then the reduction is not polynomial.) The authors pose as an open problem the question of whether one can achieve a scheme where the security of the reduction can be made polynomial both in the number of signatures obtained by the adversary before the break and in the size of the keys.

Our Result: In the next section, we formally define the notion of security of a blind digital signature scheme. Informally, a blind digital signature scheme is secure if it satisfies both a *blindness* and a *non-forgeability* property. The blindness property was formulated in the original paper of Chaum [6], and non-forgeability was considered in the paper of Pointeval and Stern [22] (where it is called called "one more" forgery). Again, informally, (see the next section for formal definitions) blindness means that a signer can not distinguish, except with negligible probability, the order in which she issued signatures, and non-forgeability means that after getting ℓ signatures, it is infeasible for the receiver to compute $\ell + 1$ signatures. We consider a non-forgeability requirement where the forger is allowed to run many parallel protocol executions for many blind signatures, in an arbitrarily interleaved and adaptive fashion, and to abort many such executions in the middle of the protocol, without having to count them as signatures. We call such an attack an *adaptive interleaved chosen-message attack*. We demonstrate a blind digital signature scheme which is secure against this attack, and which can be implemented based on any one-way trapdoor permutation.

MAIN THEOREM: *Assume that one-way trapdoor permutations exist. Then there exists polynomial-time blind digital signature scheme, secure against an adaptive interleaved chosen-message attack.*

Our scheme has both advantages and disadvantages. We list them below.

Advantages:

- We give the first complexity-theoretic proof of security for blind digital signatures; our scheme is shown to be secure against the adaptive interleaved chosen-message attack. (All previous proofs of security for blind digital signatures were in the random-oracle model only and were not fully polynomial.)
- We show how to achieve our protocol based on any one-way trapdoor permutation. (All previous blind digital signatures schemes were based on number-theoretic assumptions only).
- Our scheme and proof of security are fully polynomial in all suitable parameters, including the number of blind signatures requested before the break. (We thus resolve in the affirmative the open question posed by Pointcheval and Stern [22].)

Disadvantages:

- Our scheme, while polynomial in all suitable parameters, is inefficient. Thus, it should be viewed merely as a proof of existence which should pave the way for efficient future implementations.

Organization of the Paper: The remainder of this paper is organized as follows. In section 2, we present the definitions of blindness and security to be used in this paper. We discuss some of the complications and solutions involved in constructing a blind signature scheme in section 3. We present our blind signature scheme in section 4 and sketch a proof of its security in section 5. We conclude in section 6 with a brief discussion of the significance of our result to the area of anonymous electronic cash.

2 Definitions

In the proof and the construction of blind digital signatures, we will use the security of standard digital signatures, as defined by Goldwasser, Micali, and Rivest [15]. Hence, before we give the definition of blind digital signatures, we remind the reader of the standard signature definitions.

Signature schemes: The standard signature scheme is a triple of algorithms, $(Gen, Sign, Verify)$, where $Gen(1^k)$ is a probabilistic polynomial time key-generation algorithm, which takes as an input a security parameter 1^k and outputs a pair (pk, sk) of public and secret keys. The signing algorithm $Sign(pk, sk, m)$ is a probabilistic polynomial time algorithm which takes as an input a public key pk a secret key sk a message m to be signed and outputs a signature of a message $\sigma(m)$ as well as a new (i.e., updated) secret key sk'. (In a *memoryless* signature scheme, the secret key sk stays the same throughout.) A verification algorithm $Verify(pk, m, \sigma(m))$ is a deterministic polynomial time algorithm which takes as an input a public key pk a message m and a purported signature $\sigma(m)$ and outputs $accept/reject$. We require, of course, that for all signatures computed by first executing a key generation algorithm and then signing a sequence of messages according to the above process, the verification algorithm always output $accept$.

As mentioned above, security against the existential adaptive chosen-message attack of Goldwasser, Micali, and Rivest is the strongest known security measure for signatures [15].

Security of Signature Schemes: In this attack, an adversary A, which is a probabilistic polynomial-time machine, is given a public key pk generated by the key-generation algorithm. The adversary A can request in an adaptive fashion a polynomial number of signatures of his choice. A must then produce a valid signature on a document for which he has not yet seen a signature. If he can produce any such document/signature pair which is accepted by the verification algorithm, then the attack is successful. A signature scheme is defined to be *secure* if for all constants c, and for all probabilistic polynomial-time A, there exists a security parameter $k_{c,A}$

such that for all $k > k_{c,A}$ the probability (taken over coin-flips of the adversary) that A is successful is less then $1/k^c$.

We shall use the term *polynomially bounded* in this paper to refer to a quantity which is polynomial in the security parameter. Similarly, we shall denote by $1/poly$ the inverse of a polynomially bounded quantity.

We are now ready to give a formal definition of a blind signature scheme and its security. In the definition below, digital signatures are treated as an interactive protocols between two players: a *Signer* (who "blindly" signs a document m) and the *User* (who obtains the signature of her document m). We rely on the formalism of Interactive Turing machines, defined by Goldwasser, Micali and Rackoff [13]. The security of a blind digital signature consists of two requirements: the **blindness** property and the **non-forgeability** of additional signatures. We say the blind digital signature scheme is *secure* if it satisfies both properties, as defined below. (We remark that our non-forgeability definition follows the definition of "one-more" forgery by Pointcheval and Stern [22])

Blind Digital Signatures: A blind digital signature scheme is a four-tuple, consisting of two Interactive Turing machines *(Signer, User)* and two algorithms *(Gen, Verify)*. $Gen(1^k)$ is a probabilistic polynomial time key-generation algorithm which takes as an input a security parameter 1^k and outputs a pair (pk, sk) of public and secret keys. The *Signer(pk, sk)* and *User(pk, m)* are a pair of polynomially-bounded probabilistic Interactive Turing machines, where both machines have the following (separate) tapes: read-only input tape, write-only output tape, a read/write work tape, a read-only random tape, and two communication tapes, a read-only and a write-only tape. They are both given (on their input tapes) as a common input a pk produced by a key generation algorithm. Additionally, the *Signer* is given on her input tape a corresponding secret key sk and the *User* is given on her input tape a message m, where the length of all inputs must be polynomial in the security parameter 1^k of the key generation algorithm. The User and Signer engage in the interactive protocol of some polynomial (in the security parameter) number of rounds. At the end of this protocol the Signer outputs either *completed* or *not-completed* and the User outputs either *fail* or $\sigma(m)$. The *Verify*$(pk, m, \sigma(m))$ is a deterministic polynomial-time algorithm, which outputs *accept/reject* with the requirement that for any message m, and for all random choices of key generation algorithm, if both Signer and User follow the protocol then the Signer always outputs *completed*, and the output of the user is always *accepted* by the verification algorithm.

We now describe the security of blind signatures.

The Security of Blind Digital Signature: a blind digital signature scheme is *secure* if for all constants c, and for all probabilistic polynomial-time algorithms A,

there exists a security parameter $k_{c,A}$ such that for all $k > k_{c,A}$ the following two considerations hold:

- **Blindness property**: Let $b \in_R \{0,1\}$ (i.e. b is a random bit which is kept secret from A). A executes the following experiment (where A controls the "signer", but not the "user", and tries to predict b):
  - **(Step 1):** $(pk, sk) \leftarrow Gen(1^k)$
  - **(Step 2):** $\{m_0, m_1\} \leftarrow A(1^k, pk, sk)$ (i.e. A produces two documents, polynomial in 1^k, where $\{m_0, m_1\}$ are by convention lexicographically ordered and may even depend on pk and sk).
  - **(Step 3):** We denote by $\{m_b, m_{1-b}\}$ the same two documents $\{m_0, m_1\}$, ordered according to the value of bit b, where the value of b is hidden from A. $A(1^k, pk, sk, m_0, m_1)$ engages in two parallel (and arbitrarily interleaved) interactive protocols, the first with $User(pk, m_b)$ and the second with $User(pk, m_{1-b})$.
  - **(Step 4):** If the first User outputs on her private tape $\sigma(m_b)$ (i.e. does not output $fail$) and the second user outputs on her private tape $\sigma(m_{1-b})$ (i.e., also does not output $fail$) then A is given as an additional input $\{\sigma(m_b), \sigma(m_{1-b})\}$ ordered according to the corresponding (m_0, m_1) order. (We remark that we do not insist that this happens, and either one or both users may output $fail$)
  - **(Step 5):** A outputs a bit $\tilde{b}$ (given her view of steps 1 through 3, and if conditions are satisfied of step 4 as well).

  Then the probability, taken over the choice of b, over coin-flips of key-generation algorithm, the coin-flips of A, and (private) coin-flips of both users (from step 3), that $\tilde{b} = b$ is at most $\frac{1}{2} + \frac{1}{k^c}$.

- **Non-forgeability property**: A executes the following experiment (where A controls the "user", but not the "signer", and tries to get "one-more" signature):
  - **(Step 1):** $(pk, sk) \leftarrow Gen(1^k)$
  - **(Step 2):** $A(pk)$ engages in polynomially many (in k) adaptive, parallel and arbitrarily interleaved interactive protocols with polynomially many copies of $Signer(pk, sk)$, where A decides in an adaptive fashion when to stop. Let ℓ denote the number of executions, where the Signer outputted $completed$ in the end of Step 2.
  - **(Step 3):** A outputs a collection $\{(m_1, \sigma(m_1)), \ldots (m_j, \sigma(m_j))$ subject to the constraint the all $(m_i, \sigma(m_i))$ for $1 \leq i \leq j$ are all $accepted$ by $Verify(pk, m_i, \sigma(m_i))$.

  Then the probability, taken over coin-flips of key-generation algorithm, the coin-flips of A, and over the (private) coin-flips of the Signer, that $j > \ell$ is at most $\frac{1}{k^c}$.

Remarks on Blindness Property:

- We stress that we do not require the adversary to follow the signing protocol, nor do we require the protocol to terminate with the valid signature. Moreover, we require that the probability bound holds even if the protocol is aborted in the middle of execution.
- By standard hybrid arguments, the above definition is as general as the definition in which polynomially many signatures are obtained and then recalled, leaving A to distinguish between the last two signatures.
- Finally, we note that since the User does not have any special ID or other special identification (or else embeds such information in the message to be signed), we restrict our view to a single user program.

3 Towards Our Scheme

As mentioned in the introduction, our scheme is somewhat complicated. Instead of presenting it immediately, we shall offer a sequence of refinements which in the end yields a correct scheme. Our aim is twofold: (1) to explain why the complications in our the scheme are necessary and (2) to elaborate on the subtleties of the problem, even when using general completeness results.

Basic Ingredients: The two basic ingredients we start with are the secure signature scheme of Naor-Yung [19], and the two-party completeness theorem of Yao and Goldreich, Micali and Widgerson [26, 14]. Let us briefly recall both ingredients.

- The signature scheme of Naor-Yung is secure against existential adaptive chosen-message attack and can be built based on any one-way permutation f [19] (we remark that we do not need the result of [24] which is based on weaker assumptions since other tools in our protocol require one-way permutations anyway.)

- The two-party completeness theorem of Yao and Goldreich, Micali and Wigderson [26, 14] basically says that for any two parties A, and B, where A is given a secret input x and B is given a secret input y, and for any polynomial-time computable function $g(\cdot,\cdot)$ there exists a protocol for computing $g(x,y)$ such that nothing except the output of the function is revealed to the players. Moreover, the schemes could be easily extended to require that only one player learns $g(x,y)$, while for the other player learns nothing (i.e. all interactions are computationally indistinguishable.) Furthermore, the value of $g(x,y)$ can be learned by one of the players only as the last message of the

protocol, with the condition that if the protocol is aborted before this last message, then again no information is revealed (i.e. all interactions are computationally indistinguishable.) In fact, we use a stronger definition, used by [26, 14]: that there exists a polynomial-time *simulator* which can simulate the views of the players, even in the case of Byzantine (i.e. malicious) faults. (For details see the above references.) Furthermore, the two-party protocol can be augmented to leave part of the input of one of the players unspecified, and allow this player to set this value in an arbitrary fashion during the actual protocol execution.

A first simple idea would be to use these two general theorems in order to construct blind signatures in the following way: instead of having the User request that the Signer sign the message in the clear, engage in the two-party private protocol, at the end of which the User learns the signature of the document, and the Signer learns nothing. This "solution" suffers from several problems, which we now elaborate upon.

Problem 1: The scheme of Naor-Yung is not "memoryless", and future signatures reveal previous signatures, which violates the blindness property.

Solution to Problem 1: Goldreich [11] showed how to make any signature scheme (including the signature scheme of Naor and Yung) "memoryless" [11], using pseudo-random functions of [12]. In our setting, the key-generation algorithm can add to the secret key a seed s for pseudo-random function and add to a public key a *commitment* [18] of this seed. Then, during secure two-party computation, the Signer must generate all of her random choices (and a random tree of [19, 11]) using an agreed-upon pseudo-random function with the committed seed.

Problem 2: Let us take a closer look at the proof of security of Naor-Yung scheme [19]. Their scheme takes as its basis a tree; messages are inserted in the leaves of this tree, and a signature involves the construction of a path from the root of the tree to the appropriate leaf. Naor and Yung show that if there exists a Forger that can replace the User and forge the signature of a new document, then this Forger can be used as a subroutine to invert a one-way permutation on a random input in this tree. The key idea of their proof is to replace the Signer with an Inverter which is able to set a "trap" in this tree as follows: in order to forge a signature, the Forger must diverge from the path of previous signatures in the tree (see, for example, [19, 11, 8]), and if the Inverter can guess where in the path this divergence takes place (which she can do with $1/poly$ probability) then it can place an output of a one-way permutation at this point and force the forger to invert. The problem is that for this proof to work, the Inverter must

know all the previous signatures, in order to know where to set a "trap". But the knowledge of previous signatures on the part of the Signer is exactly what blind signatures are trying to prevent! These would seem to be contradictory requirements.

Solution to Problem 2: Since the Inverter is deployed in a simulation of the signature process, the Inverter is allowed to "reset" the Forger. So how can we assure that the Signer (who can not "reset") does not know which documents she signs while the Inverter (which is allowed to "reset") has full information? The idea is to use a variant of a proof of knowledge procedure. The User first commits to a random string r and to her message exclusive-ored with r. The Signer requests to see the decommitment of either one or the other commitment (but not both). The Inverter will be able to retrieve the message by first requesting to see one commitment, then resetting the state of the Forger, and then requesting to see the other commitment. We call such a commitment an *extractable* commitment.

We should point out that since both commitments (and their decommitments) are now part of the input (public and private) of the secure two-party completeness protocol, they are included in the execution of the two-party completeness protocol and hence force correct behavior of both players (see [26, 14]).

Problem 3: In the scheme of [11] for rendering the signature scheme memoryless, it was not necessary for the Signer to prove that she is only using coin-flips that come from a pseudo-random function. In order to achieve the blindness property, however, we must insist that this is always the case. (This is done through use of the completeness theorem in conjunction with a published commitment of the pseudo-random seed S, as we shall see.) The memoryless property of the signature guarantees that the Signer can not "mark" the signatures in any way, an absolutely necessary property for blind signatures! In the proof of security, though – i.e., when dealing with a forger – the Inverter must be able to replace a pseudo-random string by a "trap". This trap is a completely random input (on which the forger will invert with $1/poly$ probability). Again, these would seem to be contradictory requirements, since if the Signer can insert new random bits into the singing process, then it can "mark" the signature and violate blindness property.

Solution to Problem 3: Again, the ability of the Inverter to "reset" the Forger is vital to the resolution of the above somewhat paradoxical issue. The idea is again to have the Signer commit (in an *extractable* form – see above) to some poly-long string X. The Signer picks a secret input Y of the same length as X; both X and Y are used as private inputs for the secure protocol guaranteed by

the two-party completeness theorem. We modify our secure function evaluation protocol to allow the Signer to deviate from the above pseudo-random choices and insert other inputs, but only in case when $X = Y$. If $X \neq Y$ we demand that the Signer follow the protocol as before. The chances that the Signer can correctly "guess" X are negligible, so the signature scheme remains blind with overwhelming probability. On the other hand, the Inverter, by resetting the Forger, can find out what X is, set her guess Y to the same value, and then set a trap.

Problem 4: Since the definition of the non-forgeability property allows the Inverter to run many parallel sessions interleaved in an arbitrary fashion, we must be assured that it can insert a "trap" (on which the Forger will invert during forging of a "one-more signature") in a consistent manner in all the runs. The Inverter must therefore be able to specify a point in the exposed sub-tree of signatures (see [19, 11]) at which to insert her trap. But how can this be consistently specified, not knowing the order or the interleaving nature of the adversary?

Solution to Problem 4: The solution is as follows: if $X = Y$ the Signer/ Inverter can insert arbitrary values at an arbitrary point (i.e. it does not commit where to insert the trap) and thus can consistently do so during parallel interleaving sessions in the same fashion as before, i.e. consistently at some point in an exposed sub-tree of signatures (see [19, 11]) We now give details how this can be done.

Recall that we use a secure computation protocol in such a way that the User/Forger receives no information about the signature prior to the last round from the signer. We refer to this as the *atomic signature* property. Recall that the Forger may request at most a polynomial number of signatures, say $p(k)$, before producing her forgery. The Inverter therefore chooses a number r uniformly at random from $[1, p(k)]$. This specifies the interaction with the Forger in which she will try to plant her trap. The Inverter also chooses a height a of a tree uniformly at random at which to plant her trap. The Inverter specifies in interaction r that trap w will be planted at height a. Once the message m in interaction r has been specified, the Inverter may determine the node v in which she has chosen to plant her trap. With probability $1/poly$, the Inverter will have chosen to plant her trap in such a way that no previously issued signature has yet made use of the node v; thus planting of the trap will not invalidate signatures issued previous to interaction r. We say in this case that the trap choice has been successful: the Inverter plants her trap with impunity. On the other hand, if the Inverter has chosen an address for her trap such that previous signatures would be invalidated, then we say that the trap choice has been unsuccessful.

In this case, the Inverter does not plant the trap in node v. By the atomic signature property, no information about signatures has been divulged to the User/Forger in any other interaction. Therefore, the Inverter may continue to plant her trap in node v in a consistent fashion for all incomplete interactions. Since the simulation is successful with probability $1/poly$, a trap is planted as in Naor and Yung's scheme with probability $1/poly$. It follows that the Inverter causes the Forger to invert w with $1/poly$ probability.

4 The Blind Signature Scheme

We shall now assemble all of the above and describe our blind signature scheme. We shall denote by $c(z)$ the secure commitment of a string z. We shall denote by $c^*(z)$ an extractable commitment of z. (Recall from above that such a commitment reveals nothing about z to the Signer, but enables an Inverter, by rewinding a Forger, to extract z.)

The scheme works as follows. The Signer publishes $c(s)$, that is, a commitment of her secret pseudo-random key s, along with her public key pk, and the one-way permutation f used in the Naor and Yung [19] scheme. (Also made public are the pseudo-random generation function g, as well as a set of public hash functions required by the scheme of Naor and Yung.)

Each time a signature is to be issued, the Signer and User engage in a secure two-party computation. The User provides as input to the computation the message m to be signed, as well as a random string X. In addition, the User provides extractable commitments $c^*(X)$ and $c^*(m)$. Through a variation on the standard secure two-party computation protocol, these two commitments are passed in the clear to the Signer. (Recall that in the Inverter/Forger scenario, these commitments enable the Inverter, by rewinding the Forger, to learn X and m, thereby effectively circumventing the blindness of the scheme.)

The Signer provides to the secure computation (of [26, 14]) her private information as well as information respecting the trap she may wish to plant (when she plays the role of the Inverter). In particular, the Signer provides to the computation her secret signing key sk and her secret pseudorandom seed s. She also provides a string Y constituting her guess of X. Finally, the Signer provides to the computation a specification of the trap she wishes to have inserted. More precisely, the Signer specifies w, the value she wishes to have planted in the signature tree, and either a node v in a tree where she wishes to put w (in case v is already known from other sessions) or a boolean value indicating that in the current signature, on its way to the leaf, at height a in the tree at which trap w should be inserted.

The memoryless property [11] is incorporated into our our scheme as follows. The secure two-party computation protocol produces a choice of leaf in which

to insert the message m; this is computed to be the output of the pseudo-random generation function g of [12] with secret seed s and index m (truncated appropriately to yield a uniform selection of leaves). If the Signer's guess Y is successful, i.e., if $Y = X$, then the signer can deviate from $g_s(m)$ path and insert instead w at a node v as specified above. (If the current signature does not use v, no trap is planted and $g_s(m)$ is followed.) On successful completion of the protocol (i.e., if cheating during secure computation was not detected) the decodinbg of signature $\sigma(m)$ (with or without trap) is sent to the User.

5 Security of our scheme

The blindness of the scheme follows from the properties of two-party secure computation of [26, 14]. The security of the computation is violated only when the guess Y of the Signer is correct, and consequently $X = Y$. This happens with negligible probability.

It now remains to be seen that if there exists a successful Forger for this scheme, then this Forger may be used by the Inverter in a polynomial-time algorithm Q capable of inverting the one-way permutation f on an arbitrary value with probability $1/poly$.

Since the Forger makes extractable commitments of X and m, the Forger can be used by the Inverter to rewind the protocol and extract X and m. By setting $X = Y$ (which is indistinguishable for any polynomial-time Forger from the case $X \neq Y$), the Inverter can now plant a trap in a consistent manner.

When signatures are issued sequentially, therefore, by making use of its knowledge of the history of issued signatures, the Inverter may set a "trap" in exactly the way that this was done in a memoryless analog of Naor and Yung's scheme. The ability of algorithm Q to invert f now follows from the security of the memoryless version of Naor and Yung's memoryless analog [11, 19].

When signatures are issued over the course of multiple, interleaved executions of the blind signature protocol, the same "trap" may be planted consistently over many executions using the method described in Section 3 (in response to Problem 4). Thus, the Inverter remains capable of inverting with probability $1/poly$ even over interleaved protocol executions.

6 Conclusion: Anonymous Electronic Cash

As mentioned in the introduction to this paper, blind digital signatures are principally of interest to the cryptographic community for their importance in anonymous electronic cash schemes. In many of these schemes, a coin consists of a pair $(d, \sigma(d))$, where d is selected from a suitable message space, and $\sigma(d)$ represents a blind signature of d or of a digest of d.

An early example of an electronic cash scheme of this sort is a paper by Chaum, Naor, and Fiat [7]. (Their system has in fact been deployed with some additional apparatus in a real-world implementation.) Here a coin assumes the form $(d, f^{1/3}(d))$, where f is a suitable hash function, such as MD5. Computations are performed in $\mathcal{Z}_N$ for some product of primes $N = pq$, where N is published, and p and q are held in secret by the Signer (the bank). A coin is issued as follows. The User generates the value d and a random blindness factor r, and sends the quantity $r^3 f(d)$ to the Signer. The Signer computes $rf^{1/3}(d)$, and sends it to the User. On dividing out r from this last quantity, the User computes $f^{1/3}(d)$, and has therefore obtained a valid coin pair $(d, f^{1/3}(d))$. It is easy to see that the described scheme is blind. (It is, in fact, blind in an information theoretic sense.) The scheme would also appear at first glance to be secure since, given d, it is hard to compute $f^{1/3}(d)$, and vice versa. We wish to point out, however, that this rationale does not give a proof of security, and is in fact deceptive: there might nonetheless be some computationally feasible way of generating the pair of values constituting the coin *simultaneously.*

This and similar weaknesses appear to vex many implementations of anonymous digital cash. Although a proof of security of several blind digital signature schemes based on the random oracle model was given by Pointcheval and Stern [22], the current paper gives the first complexity-based proof for this important primitive. We have therefore shown that secure anonymous digital cash is possible to achieve in a complexity-based sense, i.e. we have shown that it may be as secure as, say, factoring. As mentioned above, however, our protocol is inefficient. Combining the requirements of efficiency and provable security to create a new blind digital signature scheme is an interesting open problem.

References

1. S.A. Brands. Untraceable Off-line Electronic Cash Based on Secret-key Certificates. Latin 95.
2. M. Bellare and S. Micali. "How to Sign Given Any Trapdoor Function". *STOC* 88.
3. M. Bellare and S. Goldwasser. "New Paradigms for Digital Signatures and Message Authentication Based on Non-Interactive Zero Knowledge Proofs". Crypto 89.
4. M. Bellare and P. Rogaway. "The Exact Security of Digital Signatures – How to Sign with RSA and Rabin". Eurocrypt-96.
5. R. Canetti "De-mystifying Random Oracles" CRYPTO-97 (this proceedings).
6. D. Chaum. "Blind Signatures for Untraceable Payments". Crypto-82.
7. D. Chaum, A. Fiat, and M. Naor. "Untraceable Electronic Cash", Crypto-89.
8. C. Dwork and M. Naor. "An Efficient Existentially Unforgeable Signature Scheme and its Applications". Crypto 94.

9. W. Diffie and M. Hellman. "New Directions in Cryptography". *IEEE Trans. on Inf. Theory*, IT-22, pp. 644–654, 1976.
10. A. Fiat and A. Shamir. "How to Prove Yourself: Practical Solutions of Identification and Signature Problems, CRYPTO 86.
11. O. Goldreich."Two Remarks Concerning the GMR Signature Scheme" MIT Tech. Report 715, 1986. CRYPTO 86.
12. O. Goldreich, S. Goldwasser, and S. Micali. "How to Construct Random Functions". *JASM* V. 33 No 4. (October 1986) pp. 792-807.
13. S. Goldwasser, S. Micali and C. Rackoff, "The Knowledge Complexity of Interactive Proof-Systems". *SIAM J. Comput.* 18 (1989), pp. 186-208; (also in STOC 85, pp. 291-304.)
14. O. Goldreich, S. Micali, and A. Wigderson. "How to Play Any Mental Game". Proc. of 19th STOC, pp. 218-229, 1987.
15. S. Goldwasser, S. Micali, and R. Rivest. "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks". *SIAM Journal of Computing* Vol. 17, No 2, (April 1988), pp. 281-308.
16. Goldwasser S., and R. Ostrovsky "Invariant Signatures and Non-Interactive Zero-Knowledge Proofs are Equivalent" CRYPTO 92.
17. L.C. Guillou and J.J. Quisquater. "A Practical Zero-Knowledge Protocol Fitter to Security Microprocessor Minimizing Both Transmission and Memory". EUROCRYPT 88.
18. M. Naor. "Bit Commitment Using Pseudo-Randomness". Crypto-89.
19. M. Naor and M. Yung. "Universal One-Way Hash Functions and their Cryptographic Applications". STOC 89.
20. T. Okamoto. "Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes" CRYPTO 92.
21. D. Pointcheval and J. Stern. "Security Proofs for Signature Schemes". Eurocrypt 96.
22. D. Pointcheval and J. Stern. "Provably Secure Blind Signature Schemes". Asiacrypt 96.
23. R.L. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems". Comm. ACM, Vol 21, No 2, 1978.
24. J. Rompel. "One-way Functions are Necessary and Sufficient for Secure Signatures". STOC 90.
25. C.P. Schnorr. "Efficient Identification and Signatures for Smart Cards". CRYPTO 89.
26. A. C. Yao. "How to Generate and Exchange Secrets". Proc. of 27th FOCS, 1986, pp. 162-167.

This article was processed using the LaTeX macro package with LLNCS style