On the (Im)possibility of Blind Message Authentication Codes

Michel Abdalla[1], Chanathip Namprempre[2], and Gregory Neven[1,3]

[1] Departement d'Informatique
École normale supérieure
45 Rue d'Ulm, 75230 Paris Cedex 05, France
Michel.Abdalla@ens.fr
http://www.di.ens.fr/~mabdalla
[2] Electrical Engineering Department
Thammasat University
Klong Luang, Patumtani 12121, Thailand
cnamprem@engr.tu.ac.th
http://www.engr.tu.ac.th/~cnamprem
[3] Department of Electrical Engineering
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, B-3001 Heverlee-Leuven, Belgium
Gregory.Neven@esat.kuleuven.ac.be
http://www.neven.org

Abstract. Blind signatures allow a signer to digitally sign a document without being able to glean any information about the document. In this paper, we investigate the symmetric analog of blind signatures, namely *blind message authentication codes* (blind MACs). One may hope to get the same efficiency gain from blind MAC constructions as is usually obtained when moving from asymmetric to symmetric cryptosystems. Our main result is a negative one however: we show that the natural symmetric analogs of the unforgeability and blindness requirements cannot be simultaneously satisfied. Faced with this impossibility, we show that blind MACs do exist (under the one-more RSA assumption in the random oracle model) in a more restrictive setting where users can share common state information. Our construction, however, is only meant to demonstrate the existence; it uses an underlying blind signature scheme, and hence does not achieve the desired performance benefits. The construction of an efficient blind MAC scheme in this restrictive setting is left as an open problem.

Keywords. Provable security, blind signatures, blind MACs.

1 Introduction

The concept. Blind signatures [7, 8] allow a signer to digitally sign a document while preventing the signer from seeing the content of the document, or even from recognizing the signature when faced with it later on. Blind signatures form a crucial anonymity-providing ingredient in digital cash protocols [7,

9], and have also been applied in a number of electronic voting schemes [7, 10, 13] to protect voters' privacy. Since their first introduction in 1982, blind signatures have become a well-studied primitive with formal security notions [12, 15], practical schemes realizing these notions under various assumptions [8, 15, 2, 5, 6], and a theoretical construction based on the existence of trapdoor one-way permutations [12].

In the same way that message authentication codes (MACs) can be seen as the symmetric-key equivalent of digital signatures, Pinkas [14] suggested *blind MACs* as the symmetric analog of blind signatures – leaving the construction of such schemes as an open problem however. In a blind MAC scheme, a user interacts with a tagger that knows a secret key K to obtain a valid tag τ for a message M, but without leaking any information about M to the tagger in the process. At a later point in time, the tagger can use K to check the validity of a given message-tag pair (M, τ), but cannot link it back to the session during which the tag was created.

Motivation. The main motivation for blind MACs is efficiency. As is the case for standard MACs, one could hope to construct blind MACs from purely symmetric primitives, so that they can provide a more efficient alternative in applications where not all of the properties provided by digital signatures are needed. Good candidate applications are those where signatures are verified by the same entity that created them. In particular, we are interested in applications in which a "signer" does not need to convince others that it has generated (or has approved) the data in question, but only needs to convince itself at some later time that the data have not been modified. In other words, only the integrity of the data, not the non-repudiation of the data source, is of interest.

The first mention in public literature (to the best of our knowledge) of blind MACs was made by Pinkas [14] in the context of a fairness-providing transformation of Yao's secure two-party computation protocol [17]. The evaluator of the circuit commits to a number of 0 and 1-bits, and has these commitments blindly signed by the circuit constructor. The constructor then puts the blinded signatures in the output tables of the garbled circuit. At the end of the protocol, the constructor and evaluator gradually open their commitments. The constructor can verify that he indeed signed the commitments being opened by the evaluator, which prevents the latter from opening a commitment to some random value instead of the real output. Pinkas noted that, since the signatures are generated and verified by the same party, blind MACs could be used instead of blind signatures. He did not provide any formal definitions of the concept however, and left an actual construction as an open problem.

Blind MACs could also be used in Chaum's original online digital cash protocol [7]. A coin in this protocol is essentially a unique identifying string that is blindly signed by the bank. When the coin is spent, the merchant verifies the bank's signature and forwards the coin to the bank. The bank checks the validity of the signature again, and looks up in a database whether the coin is being double-spent. If not, it transfers the correct amount to the merchant's account, and adds the coin's identifying string to the database. Since the bank has to be

online at the time the coin is spent anyway, the merchant may just as well leave the verification entirely up to the bank, so that the latter can use blind MACs instead of blind signatures (assuming that the clients' bank is the same as the merchants' bank). The gain in efficiency will reduce the infrastructural requirements brought about by online payment processing, and may actually make the protocol feasible in practice. In fact, we recently learned that blind MACs were already considered in this particular context by the Digicash research team [16]. They did not further pursue this idea because they suspected blind MACs to be impossible, without proving this fact however.

A third instance where blind MACs could take the place of blind signatures is in certain electronic voting schemes. The protocol of Fujioka et al. [10] for example works as follows. Voters commit to their votes, and have the commitment blindly signed by an administrator who checks their right to vote. All voters then send the signed commitment through an anonymous channel to a second authority called the counter. The counter verifies the administrator's signature and publishes all commitments on a bulletin board. At the end of the voting stage, each voter checks that his/her vote is posted on the bulletin board, and publicly complains if it is not. Finally, voters anonymously send the opening information for their commitments to the counter, who publishes everything on the bulletin board and announces the result of the election.

Note that in this protocol, the signer and verifier are not the same entity. Nevertheless, the administrator could use a blind MAC scheme to tag the voters' commitments, and reveal the tagging key after the end of the voting stage. MAC values are more efficiently verified than signatures, thus lowering the computational threshold for citizens to perform an independent audit of the election. A disadvantage is that the counter cannot verify the validity of commitments before posting them on the bulletin board, possibly resulting in more "junk" votes being published there. This problem however is also present in the original scheme, since voters can publish false complaints, of which the validity has to be checked as well. Moreover, if the counter can be trusted not to create fake registrations, then the administrator could give him the secret key at the start of the election already, allowing him to "weed out" junk votes earlier on.

Our results. We first give proper definitions for the syntax and security of blind MACs, modeled after those of blind signatures. Our main result is a negative one: in Theorem 4, we show that the natural symmetric analogs of the one-more unforgeability [15] and blindness [12] requirements are contradictory, meaning that blind MACs satisfying both properties simultaneously cannot exist. Intuitively, the problem is that, because of the absence of a public key, the user has no way to check whether the tagger is using the same key throughout different tagging sessions. We present a universal adversary that breaks the blindness of any blind MAC scheme by using different keys in different tagging sessions, and we show that this attacker always succeeds, unless the scheme is forgeable.

Faced with the impossibility of blind MACs in their most general definition, we investigate whether they can exist under a more restrictive, yet still somewhat

useful definition. In Section 5, we give a provably secure blind MAC construction in a setting where users share common state information. Whether this setting is realistic depends on the application. For Pinkas' two-party computation protocol [14], this is a perfectly reasonable assumption since there is only one user, the circuit evaluator, who can easily maintain state throughout different signing sessions. For digital cash and voting schemes however, it may be less realistic to assume the availability of common state information.

The sole purpose of our construction is to demonstrate the existence of blind MACs in this restrictive setting. It is based on an underlying blind signature scheme, and therefore does *not* achieve the performance benefits one would hope to get from a blind MAC scheme. We argue however that, before trying to come up with efficient constructions, it is important to understand what it exactly is that we are trying to construct, and whether it can be constructed at all. The fact that blind MACs can be constructed from blind signatures may sound rather unsurprising at first, but is not trivial: firstly, our impossibility result shows that not even such a "trivial" construction exists in the most natural definition of blind MACs, and secondly, our construction needs a special form of blindness from the underlying blind signature scheme, which we had to prove to be satisfied by a slight variant of Chaum's scheme [8].

Organization. Section 2 recalls the definition of blind signatures and the security notions. Section 3 presents the definition and security notions for blind MACs. Section 4 states and proves the impossibility result. Section 5 describes the weaker model with state-sharing users, and shows that a secure blind MAC scheme exists in this model. Section 6 considers extensions to concurrent attack scenarios. Section 7 lists a few open problems.

2 Blind Signatures

Notation. We let $\mathbb{N} = \{1, 2, 3, \ldots\}$ denote the set of natural numbers. If $k \in \mathbb{N}$, then 1^k is the string of k ones. The empty string is denoted ε. If x, y are strings, then $|x|$ is the length of x and $x\|y$ is the concatenation of x and y. If S is a set, then $|S|$ is its cardinality. If A is a randomized algorithm, then $\mathsf{A}(x_1, x_2, \ldots : \mathrm{O}_1, \mathrm{O}_2, \ldots)$ means that A has inputs $x_1, x_2, \ldots$ and access to oracles $\mathrm{O}_1, \mathrm{O}_2, \ldots$. Also $y \xleftarrow{\$} \mathsf{A}(x_1, x_2, \ldots : \mathrm{O}_1, \mathrm{O}_2, \ldots)$ means that we run the randomized algorithm A on inputs $x_1, x_2, \ldots$ and with access to oracles $\mathrm{O}_1, \mathrm{O}_2, \ldots$, and let y denote the output obtained.

An *interactive algorithm* is a stateful algorithm that on input an incoming message M_{in} (this is ε if the party is initiating the protocol) and state information St outputs an outgoing message M_{out} and updated state St'. For an interactive algorithm A having access to oracles $\mathrm{O}_1, \mathrm{O}_2, \ldots$, this is written as $(M_{\text{out}}, St') \xleftarrow{\$} \mathsf{A}(M_{\text{in}}, St : \mathrm{O}_1, \mathrm{O}_2, \ldots)$. Two interactive algorithms A and B are said to *interact* when the outgoing messages of A are passed as incoming messages to B, and vice versa, until both algorithms enter either the `halt` or the `fail` state. We write $(M_{\mathsf{A}}, St_{\mathsf{A}}, M_{\mathsf{B}}, St_{\mathsf{B}}) \xleftarrow{\$} [\mathsf{A}(St_{\mathsf{A}}) \leftrightarrow \mathsf{B}(St_{\mathsf{B}})]$ to denote the final outgoing messages and states after an interaction between A and B when run on initial

states St_{A} and St_{B}, respectively. More formally, it is the outcome of the following experiment:

$M_{\mathsf{B}} \leftarrow \varepsilon$
Repeat
  $(M_{\mathsf{A}}, St_{\mathsf{A}}) \xleftarrow{\$} \mathsf{A}(M_{\mathsf{B}}, St_{\mathsf{A}})$; $(M_{\mathsf{B}}, St_{\mathsf{B}}) \xleftarrow{\$} \mathsf{B}(M_{\mathsf{A}}, St_{\mathsf{B}})$
Until $\{St_{\mathsf{A}}, St_{\mathsf{B}}\} \subseteq \{\texttt{halt}, \texttt{fail}\}$
Return $(M_{\mathsf{A}}, St_{\mathsf{A}}, M_{\mathsf{B}}, St_{\mathsf{B}})$

Syntax of blind signatures. We repeat the definition of blind signatures as proposed by Juels et al. [12]. A blind signature scheme $\mathcal{BS}$ is a tuple of four polynomial-time algorithms $(\mathsf{Kg}, \mathsf{User}, \mathsf{Sign}, \mathsf{Vf})$ where

- the randomized *key generation* algorithm Kg, on input a security parameter 1^k with $k \in \mathbb{N}$, outputs a public key pk and a corresponding secret key sk.
- User and Sign are possibly randomized interactive algorithms called the *user* and *signer* algorithm, respectively. The user runs the User algorithm on an initial state consisting of a public key pk and a message $M \in \{0,1\}^*$, and lets it interact with the Sign algorithm that is run by the signer on initial state a secret key sk. At the end of the protocol, the User algorithm either enters the `halt` state and outputs a signature σ as its last outgoing message, or enters the `fail` state to indicate failure. The Sign algorithm simply enters the `halt` state at the end of the protocol, without generating any output.
- the deterministic *verification* algorithm Vf takes a public key pk, a message $M \in \{0,1\}^*$ and a signature σ as input, and outputs `acc` or `rej` to indicate acceptance or rejection of the signature, respectively.

Correctness of a blind signature scheme requires that for all $k \in \mathbb{N}$ and for all $M \in \{0,1\}^*$, it holds that $St_{\mathsf{User}} = \texttt{halt}$ and $\mathsf{Vf}(pk, M, \sigma) = \texttt{acc}$ when $(pk, sk) \xleftarrow{\$} \mathsf{Kg}(1^k)$ and $(M_{\mathsf{Sign}}, St_{\mathsf{Sign}}, \sigma, St_{\mathsf{User}}) \xleftarrow{\$} [\mathsf{Sign}(sk) \leftrightarrow \mathsf{User}((pk, M))]$ with probability 1.

Unforgeability of blind signatures. The security of a blind signature scheme is twofold: on the one hand, a user should not be able to forge signatures (*unforgeability*), and on the other hand, the signer should not be able to see the message that is being signed, or even be able to relate signed messages to previous protocol sessions (*blindness*).

The standard definition of existential unforgeability under chosen-message attack [11] does not apply to blind signatures: the signer doesn't see the messages he signs, and hence the experiment has no way of telling whether the forgery is on a new message or on a message that was signed before. Therefore, we use the notion of *one-more unforgeability* as introduced by Pointcheval and Stern [15]. Let $\mathcal{BS} = (\mathsf{Kg}, \mathsf{User}, \mathsf{Sign}, \mathsf{Vf})$ be a blind signature scheme, let $k \in \mathbb{N}$ be the security parameter, and let A be a forging algorithm. The experiment first generates a fresh key pair $(pk, sk) \xleftarrow{\$} \mathsf{Kg}(1^k)$, and runs A on input $(1^k, pk)$. The adversary has access to a signing oracle that runs the $\mathsf{Sign}(sk, \cdot)$ algorithm and maintains state across invocations. (In a *sequential attack*, only

one signing session can be active at the same time, while a *parallel attack* allows arbitrarily interleaved sessions. For simplicity, we concentrate on sequential attacks first, and postpone the discussion of parallel attacks to Section 6.) At the end of its execution, the adversary outputs a set of message-signature pairs $\{(M_1, \sigma_1), \ldots, (M_m, \sigma_m)\}$. Let n be the number of completed signing sessions during A's attack. Then A is said to win the game if $\mathsf{Vf}(pk, M_i, \sigma_i) = \mathtt{acc}$ for all $1 \leq i \leq m$, all M_i are different and $m > n$.

The advantage function $\mathbf{Adv}^{\text{omu-sa}}_{\mathcal{BS},\mathsf{A}}(k)$ is defined as A's probability of winning the above game, and $\mathcal{BS}$ is said to be *one-more unforgeable under sequential attacks* (omu-sa-secure) if this is a negligible function for all polynomial-time adversaries A. We note here that, in the definition above and in the rest of the paper, the "time complexity" is the worst case total execution time of the experiment plus the code size of the adversary in some fixed RAM model of computation.

Blindness of blind signatures. We present a sequential variant of the blindness notion as introduced by Juels et al. [12]. The adversary now plays the role of a cheating signer, who is trying to distinguish between two signatures created in different signing sessions. The experiment chooses a random bit b, generates a fresh key pair (pk, sk) and runs the adversary A on input $(1^k, pk, sk)$. The adversary outputs two challenge messages M_0 and M_1. Then, the adversary plays the role of the signer in two sequential interactions with a User algorithm. If $b = 0$, then the first interaction is with $\mathsf{User}(pk, M_0)$ and the second is with $\mathsf{User}(pk, M_1)$; if $b = 1$, then A first interacts with $\mathsf{User}(pk, M_1)$ and then with $\mathsf{User}(pk, M_0)$. If in both sessions the User algorithms accept, then A is additionally given the resulting signatures σ_0, σ_1 for messages M_0, M_1. The adversary outputs its guess d and wins the game if $b = d$. The advantage $\mathbf{Adv}^{\text{blind-sa}}_{\mathcal{BS},\mathsf{A}}(k)$ is defined as $2p - 1$, where p is the probability that A wins this game. The scheme $\mathcal{BS}$ is said to be *blind under sequential attacks* (blind-sa-secure) if this is a negligible function for all polynomial-time adversaries A. We refer to the full version [1] for formal descriptions of the experiments defining security for blind signatures.

3 Blind MACs

Syntax of blind MACs. We define the syntax and security of blind MAC schemes in analogy to those of blind signatures.

Definition 1 [**Syntax of a blind MAC scheme.**] A blind MAC scheme $\mathcal{BMAC}$ is a tuple of four polynomial-time algorithms $(\mathsf{Kg}, \mathsf{User}, \mathsf{Tag}, \mathsf{Vf})$ where

- the randomized *key generation* algorithm Kg, on input a security parameter 1^k with $k \in \mathbb{N}$, outputs a key K.
- User and Tag are possibly randomized interactive algorithms called the *user* and *tagging* algorithm, respectively. The user runs the User algorithm on an initial state containing the security parameter 1^k and a message $M \in \{0,1\}^*$,

and lets it interact with the Tag algorithm that is run by the tagger on initial state the key K. [4] At the end of the protocol, the User algorithm either enters the `halt` state and outputs a MAC value τ as its outgoing message, or enters the `fail` state to indicate failure. The Tag algorithm simply enters the `halt` state at the end of the protocol, without generating any output.

– the deterministic *verification* algorithm Vf takes a key K, a message $M \in \{0,1\}^*$ and a MAC value τ as input, and outputs `acc` or `rej` to indicate acceptance or rejection of the MAC value, respectively.

Correctness of a blind MAC scheme requires that for all $k \in \mathbb{N}$ and for all $M \in \{0,1\}^*$, with probability 1 it holds that $St_{\mathsf{User}} = \texttt{halt}$ and $\mathsf{Vf}(K, M, \tau) = \texttt{acc}$ whenever $K \xleftarrow{\$} \mathsf{Kg}(1^k)$ and $(M_{\mathsf{Tag}}, St_{\mathsf{Tag}}, \tau, St_{\mathsf{User}}) \xleftarrow{\$} [\mathsf{Tag}(K) \leftrightarrow \mathsf{User}((1^k, M)]$.

Security of blind MACs. Analogously to the security of blind signatures, the security of a blind MAC scheme consists of an unforgeability and a blindness requirement. The game defining unforgeability works as follows. The experiment generates a fresh key $K \xleftarrow{\$} \mathsf{Kg}(1^k)$, and runs the adversary A on input 1^k. The adversary can interact in sequential sessions with a tagging oracle that runs the Tag algorithm initialized with key K. At the end of its execution, A outputs m message-tag pairs. The adversary wins the game if all messages are different, all tags are valid under key K, and $m > n$, where n is the number of completed tagging sessions during the attack. We give a more formal description of the definition below.

Definition 2 [**Unforgeability of a blind MAC scheme.**] Let $\mathcal{BMAC} = (\mathsf{Kg}, \mathsf{User}, \mathsf{Tag}, \mathsf{Vf})$ be a blind message authentication scheme. Let $k \in \mathbb{N}$, and let A be a forger with access to the tagging oracle. Consider the following experiment.

Experiment $\mathbf{Exp}^{\text{omu-sa}}_{\mathcal{BMAC},\mathsf{A}}(k)$:
$K \xleftarrow{\$} \mathsf{Kg}(1^k)$; $n \leftarrow 0$
$\{(M_1, \tau_1), \ldots, (M_m, \tau_m)\} \xleftarrow{\$} \mathsf{A}(1^k : \textsc{Tag}(\cdot))$
If $\mathsf{Vf}(K, M_i, \tau_i) = \texttt{acc}$ for all $1 \le i \le m$
and $m > n$ and $M_i \neq M_j$ for all $1 \le i < j \le m$
then return 1 else return 0,

where A's queries to the tagging oracle are answered as follows:

Oracle $\textsc{Tag}(M_{\text{in}})$:
If $M_{\text{in}} = \perp$ then $St_{\mathsf{Tag}} \leftarrow K$; $M_{\text{out}} \leftarrow \perp$
else $(M_{\text{out}}, St_{\mathsf{Tag}}) \xleftarrow{\$} \mathsf{Tag}(M_{\text{in}}, St_{\mathsf{Tag}}[s])$
If $St_{\mathsf{Tag}} = \texttt{halt}$ then $n \leftarrow n + 1$
Return M_{out}

[4] We need to pass 1^k as a parameter to the User algorithm, because otherwise it would no longer be a polynomial-time algorithm if the message is of logarithmic length. Moreover, since the user does not know the key itself, it is reasonable to give it 1^k so that at least it can check whether the tagger is using a key of the correct size.

The omu-sa advantage of A in breaking $\mathcal{BMAC}$ is defined as the probability that the above experiment returns 1:

$$\mathbf{Adv}^{\text{omu-sa}}_{\mathcal{BMAC},\mathsf{A}}(k) \;=\; \Pr\left[\,\mathbf{Exp}^{\text{omu-sa}}_{\mathcal{BMAC},\mathsf{A}}(k) = 1\,\right],$$

and $\mathcal{BMAC}$ is said to be *one-more unforgeable under sequential attacks* (omu-sa-secure) if the advantage $\mathbf{Adv}^{\text{omu-sa}}_{\mathcal{BMAC},\mathsf{A}}(k)$ is a negligible function in the security parameter k for all adversaries A with time complexity polynomial in k. ▌

In the blindness game, the experiment chooses a random bit b and generates a fresh key $K \xleftarrow{\$} \mathsf{Kg}(1^k)$. On input $(1^k, K)$, the adversary A first outputs two messages M_0, M_1. The adversary then sequentially interacts with two User sessions, playing the role of the tagger. If $b = 0$, then the first user session is initialized with message M_0, and the second with M_1; if $b = 1$, then the first session is initialized with message M_1, and the second with M_0. If both User algorithms accept, the adversary gets to see both resulting tags τ_0, τ_1 for messages M_0, M_1. The adversary has to guess the value of b.

We stress that the experiment does not enforce the resulting tags to be valid under key K. While we could include such restriction in the formal security notion, it would be out of touch with reality: the secret key K is not known to the users, so there is nobody to enforce this restriction in the real world. In fact, as we will see in the next section, it is exactly this lack of verifiability of tags that plays a central role in the proof of impossibility of blind MACs. We give a formal blindness definition below.

Definition 3 [**Blindness of a blind MAC scheme.**] Let $\mathcal{BMAC} = (\mathsf{Kg}, \mathsf{User}, \mathsf{Tag}, \mathsf{Vf})$ be a blind message authentication scheme. Let $k \in \mathbb{N}$, and let A be an adversary. Consider the following experiment.

Experiment $\mathbf{Exp}^{\text{blind-sa}}_{\mathcal{BMAC},\mathsf{A}}(k)$:
$b \xleftarrow{\$} \{0,1\}$; $K \xleftarrow{\$} \mathsf{Kg}(1^k)$
$((M_0, M_1), St_{\mathsf{A}}) \xleftarrow{\$} \mathsf{A}(\varepsilon, (1^k, K))$
$(M_{\mathsf{A}}, St_{\mathsf{A}}, \tau_b, St_b) \xleftarrow{\$} [\mathsf{A}(St_{\mathsf{A}}) \leftrightarrow \mathsf{User}((1^k, M_b))]$
$(M_{\mathsf{A}}, St_{\mathsf{A}}, \tau_{1-b}, St_{1-b}) \xleftarrow{\$} [\mathsf{A}(St_{\mathsf{A}}) \leftrightarrow \mathsf{User}((1^k, M_{1-b}))]$
If $St_0 = \texttt{fail}$ or $St_1 = \texttt{fail}$ then $\tau \leftarrow \texttt{fail}$ else $\tau \leftarrow (\tau_0, \tau_1)$
$d \xleftarrow{\$} \mathsf{A}(\tau, St_{\mathsf{A}})$
If $b = d$ then return 1 else return 0

The blind-sa advantage of A in breaking $\mathcal{BMAC}$ is defined as

$$\mathbf{Adv}^{\text{blind-sa}}_{\mathcal{BMAC},\mathsf{A}}(k) \;=\; 2 \cdot \Pr\left[\,\mathbf{Exp}^{\text{blind-sa}}_{\mathcal{BMAC},\mathsf{A}}(k) = 1\,\right] - 1$$

and $\mathcal{BMAC}$ is said to be *blind under sequential attacks* (blind-sa-secure) if the advantage $\mathbf{Adv}^{\text{blind-sa}}_{\mathcal{BMAC},\mathsf{A}}(k)$ is a negligible function in the security parameter k for all adversaries A with time complexity polynomial in k. ▌

4 Impossibility of Blind MACs

In this section, we show that blind MAC schemes simultaneously satisfying the one-more unforgeability and blindness requirements cannot exist. We do so by demonstrating a universal blindness adversary A and a universal forger F so that for any candidate scheme, one of them always has a non-negligible chance of success.

Theorem 4 [**Secure blind MAC schemes do not exist.**] Let $\mathcal{BMAC}$ be a blind MAC scheme. Either $\mathcal{BMAC}$ is one-more forgeable under sequential attacks, or it is not blind under sequential attacks.

Proof (Theorem 4). We define an adversary A breaking the blindness of $\mathcal{BMAC}$ and an adversary F breaking the one-more unforgeability of $\mathcal{BMAC}$, both under sequential attacks, so that

$$\mathbf{Adv}^{\text{blind-sa}}_{\mathcal{BMAC},\mathsf{A}}(k) + \mathbf{Adv}^{\text{omu-sa}}_{\mathcal{BMAC},\mathsf{F}}(k) = 1 \ ,$$

from which the theorem follows.

The key idea in constructing A is from the observation that, in a blind MAC scheme, the user has no way of telling under which key a tag is computed. Our adversary exploits this fact by using two different keys to generate the tags for the two user sessions. Then, it only needs validate one of the final message-tag pairs to determine during which user session the tag was computed. The possibility that a tag computed with the second key is also valid under the first key, or that both keys happen to be identical, is ruled out by the existence of a forger F that is successful in exactly these cases.

We now present both adversaries in more detail. Algorithm A, on initial state $(1^k, K)$, generates a second key $K' \xleftarrow{\$} \mathsf{Kg}(1^k)$ and outputs challenge messages $M_0 = 0$ and $M_1 = 1$. (In fact, any two distinct challenge messages would do.) It interacts with the first User algorithm by honestly running $\mathsf{Tag}(K)$, and with the second by running $\mathsf{Tag}(K')$. Since both K and K' are keys generated by the Kg algorithm, the correctness requirement for $\mathcal{BMAC}$ implies that neither of the user sessions fails, and hence that A gets back tags (τ_0, τ_1). If $\mathsf{Vf}(K, M_0, \tau_0) = \mathtt{acc}$, the adversary returns $d = 0$, else it returns $d = 1$.

The forger F works as follows: on input 1^k, it generates a fresh random key $K' \xleftarrow{\$} \mathsf{Kg}(1^k)$. It simulates an interaction $(M, St_{\mathsf{Tag}}, \tau, St_{\mathsf{User}}) \xleftarrow{\$} [\mathsf{Tag}(K') \leftrightarrow \mathsf{User}((1^k, M_0))]$ in which a tagger uses key K' to tag message $M_0 = 0$ (or whichever message M_0 algorithm A used above). It then outputs $\{(M_0, \tau)\}$ as its single forgery without making any tagging oracle call.

Now, we analyze the success probability of A and F. From Definition 3,

$$\begin{aligned}
&\mathbf{Adv}^{\text{blind-sa}}_{\mathcal{BMAC},\mathsf{A}}(k) = 2 \cdot \Pr\left[\, \mathbf{Exp}^{\text{blind-sa}}_{\mathcal{BMAC},\mathsf{A}}(k) = 1 \,\right] - 1 \\
&= \Pr\left[\, \mathbf{Exp}^{\text{blind-sa}}_{\mathcal{BMAC},\mathsf{A}}(k) = 1 \mid b = 1 \,\right] + \Pr\left[\, \mathbf{Exp}^{\text{blind-sa}}_{\mathcal{BMAC},\mathsf{A}}(k) = 1 \mid b = 0 \,\right] - 1 \\
&= \Pr\left[\, \mathbf{Exp}^{\text{blind-sa}}_{\mathcal{BMAC},\mathsf{A}}(k) = 1 \mid b = 1 \,\right] \\
&= 1 - \mathbf{Adv}^{\text{omu-sa}}_{\mathcal{BMAC},\mathsf{F}}(k)
\end{aligned}$$

The second equality follows easily from simple algebra and the fact that b is a randomly chosen bit. The third equality follows from the correctness requirement of $\mathcal{BMAC}$: if $b = 0$, then during the first user session, A tags message M_0 with key K. Hence, verification of the resulting tag with key K must always succeed, making A output the correct guess $d = 0$ with probability 1. Looking closely at the probability on the third line, we see that it is one minus the probability that a tag τ_0 obtained from an interaction $[\mathsf{Tag}(K') \leftrightarrow \mathsf{User}((1^k, M_0))]$ also verifies correctly under an independently generated key K. This however is exactly the success probability of our forger F, leading to the last equation, which concludes the proof.

5 Blind MACs for State-Sharing Users

The attack in Section 4 is due to the fact that, unlike in the case of blind signatures, the user has no public key based on which it can check whether the tagger is behaving honestly, and in particular, whether he's using the correct key to tag the message. The attack only holds however for user sessions that are completely isolated from each other, and does not exclude the existence of blind MACs when user sessions can communicate with one another. Depending on the application, it may be unrealistic to assume that all users are connected through secure communication channels (or even know of each other's existence), but it may be more reasonable to assume that small groups of user sessions can share some common state information. We ask ourselves whether a weaker form of blindness is achievable, where anonymity is guaranteed among messages tagged in state-sharing user sessions. For applications like electronic cash and voting, this would provide a rather limited form of anonymity. In Pinkas' two-party computation protocol however [14], there is only one user (the circuit evaluator), so it is perfectly safe to assume that the different user sessions share common state information.

In the following, we describe a provably secure construction of a blind MAC scheme in the state-sharing users setting. The main purpose of the construction, however, is to prove the existence of blind MACs in this restrictive setting: it is based on an underlying blind signature scheme, and hence does not achieve the performance benefits that were the original motivation for blind MACs. The secret key of the blind MAC scheme contains both the public and the private key of the underlying blind signature scheme. In the first move of the tagging protocol, the tagger sends the public key to the user. If the common state information is empty, then the user stores this public key in the common state information; otherwise, the user compares the public key to the one that is stored in the common state, and rejects if the keys are different. The rest of the protocol is identical to that of the blind signature scheme. To prove the security of the construction, we introduce a new (and actually, more natural) blindness notion for blind signatures that we call *dishonest-key blindness*, where the public key can be maliciously constructed by the adversary, rather than being honestly generated through the key generation algorithm. Then, we show

that Chaum's blind signature scheme with a prime encryption exponent whose value is larger than the RSA modulus is (unconditionally) dishonest-key blind. Together with the known fact that this scheme is one-more unforgeable in the random oracle model under the one-more RSA assumption [4], this implies that a blind MAC scheme with state-sharing users exists in the random oracle model if the one-more RSA-inversion problem [4] is hard.

Syntax and security of blind MACs with state-sharing users. We model the common state information as a third input string CSt that is given to the User algorithm, and that the user can update through a third output string CSt'. We add this common state as an input to the user in the blindness experiment in Definition 1. The common state is initialized to ε and maintained between both user sessions. The rest of the experiment remains the same. The unforgeability notion as stated in Definition 2 remains unchanged.

A construction based on blind signatures. The main idea for our blind MAC construction is to store the public key for the base blind signature scheme in the users' common state information. Then, we use the algorithms of the blind signature scheme in a natural way.

Construction 5 [**A blind MAC scheme for state-sharing users.**] Let $\mathcal{BS} = (\mathsf{Kg}_s, \mathsf{User}_s, \mathsf{Sign}, \mathsf{Vf}_s)$ be a blind signature scheme. We associate to it a blind MAC scheme $\mathcal{BMAC} = (\mathsf{Kg}_m, \mathsf{User}_m, \mathsf{Tag}, \mathsf{Vf}_m)$ as follows:

- On input 1^k, the key generation algorithm Kg_m runs $\mathsf{Kg}_s(1^k)$ to obtain a key pair (pk, sk), sets $K \leftarrow (pk, sk)$ and returns K.
- On input K, the tagging algorithm Tag starts the interaction with User_m by parsing K as (pk, sk), sends pk to User_m, runs Sign on initial state sk interacting with User_m to completion. It sets its state to whatever Sign does.
- On inputs an initial state 1^k, a message M, and an initial shared-state CSt, the algorithm User_m first receives pk from Tag. If $CSt = \varepsilon$, then User_m sets $CSt \leftarrow pk$. Otherwise, it sets $pk \leftarrow CSt$ and runs User_s on the initial state (pk, M) interacting with Tag until the interaction completes. It sets its state and output to those of User_s.
- On input a key K, a message M, and a MAC value τ, the algorithm Vf_m parses K as (pk, sk), and returns $\mathsf{Vf}_s(pk, M, \tau)$. ▌

Dishonest-key blindness for blind signatures. Before stating the security of our blind MAC construction, we briefly describe here the concept of dishonest-key blindness, which is needed to prove its security. Recall that the standard blindness notion for blind signatures assumes that the adversary is given a key pair generated properly through the key generation algorithm. This however does not cover attacks where the signer creates a public key in a special, malicious way that allows him to break the blindness of the scheme. The dishonest-key blindness notion that we propose gives the adversary more power by letting it dictate the public key to be used. This public key need not be generated by the Kg_s algorithm, nor does the adversary need to know the corresponding secret key. The adversary gets as only input 1^k, and outputs challenge messages

M_0, M_1 along with the public key pk. The rest of the experiment is unchanged: the adversary engages in two sequential User sessions that are initialized with $(1^k, pk, M_0)$ and $(1^k, pk, M_1)$, the order depending on the experiment's choice for bit b.

It is based on this stronger security requirement of the underlying blind signature scheme that we construct a secure blind MAC in the state-sharing model. Let $\mathbf{Adv}^{\text{dk-blind-sa}}_{\mathcal{BS},\mathsf{A}}(k)$ be the advantage of an adversary A in winning the above game against $\mathcal{BS}$ in a sequential attack. We say that $\mathcal{BS}$ is dk-blind-sa-secure if this advantage is a negligible function in k for all polynomial-time algorithms A. We refer to Appendix A for a formal definition of dishonest-key blindness.

SECURITY. The following theorem states that, if the underlying blind signature scheme is one-more unforgeable and *dishonest-key blind*, then the resulting blind MAC scheme is secure.

Theorem 6 If a blind signature scheme $\mathcal{BS}$ is one-more unforgeable and dishonest-key blind under sequential attacks, then the blind MAC scheme with state-sharing users $\mathcal{BMAC}$ associated to $\mathcal{BS}$ as per Construction 5 is one-more unforgeable and blind under sequential attacks.

Theorem 6 follows directly from the following two lemmas.

Lemma 7 If a blind signature scheme $\mathcal{BS}$ is omu-sa secure, then the blind MAC scheme with state-sharing users $\mathcal{BMAC}$ associated to $\mathcal{BS}$ as per Construction 5 is also omu-sa secure.

Lemma 8 If a blind signature scheme $\mathcal{BS}$ is dk-blind-sa secure, then the blind MAC scheme with state-sharing users $\mathcal{BMAC}$ associated to $\mathcal{BS}$ as per Construction 5 is blind-sa-secure.

Proof (Lemma 7). We prove the lemma via a standard reduction, namely, we assume the existence of a forger F_m mounting an attack against $\mathcal{BMAC}$, and construct a forger F_s mounting an attack against $\mathcal{BS}$ so that, if the success probability of the former is non-negligible, then so is that of the latter. The idea is for F_s to run F_m using its signing oracle to simulate F_m's tagging oracle $\textsc{Tag}(\cdot)$. Since the only difference between a tagger-user interaction in $\mathcal{BMAC}$ and a signer-user interaction in $\mathcal{BS}$ is in the public key that the tagger sends to the user as the first message, this simulation can be done perfectly. Thus, if F_m is able to produce one more valid message-tag pair than the number of finished interactive sessions with its tagging oracle, then so can F_s with respect to its signing oracle.

Now we provide more details of how F_s works. Let $\mathcal{BS} = (\mathsf{Kg}_s, \mathsf{User}_s, \mathsf{Sign}, \mathsf{Vf}_s)$ and let $\mathcal{BMAC} = (\mathsf{Kg}_m, \mathsf{User}_m, \mathsf{Tag}, \mathsf{Vf}_m)$. On input $(1^k, pk)$, it runs $\mathsf{F}_m(1^k)$. For each tagging session that F_m runs, F_s starts the interaction by sending pk to F_m as the first message, then simply relays messages between F_m and its own signing oracle. When F_m eventually halts, F_s outputs whatever F_m does.

Forger F_s perfectly simulates the environment for F_m. To see this, let pk be F_s's input public key, and let sk be the matching secret key used by its signing oracle. Notice that from the definition of $\mathcal{BMAC}$ in Construction 5, each interaction in the transcript of messages between the tagger $\mathsf{Tag}(pk, sk)$ and a user $\mathsf{User}_m(1^k, M)$ is composed of pk followed by other messages generated through the interaction between the signer $\mathsf{Sign}(sk)$ and $\mathsf{User}_s(pk, M)$ for any message M. Since all F_s does is to first send pk and then to relay messages between the signing oracle and F_m (who is acting in the role of User_m), F_s simulates F_m in the exact same environment as that of the experiment in Definition 2.

Furthermore, let $(M_1, \sigma_1), \ldots, (M_m, \sigma_m)$ be the outputs of F_s. By definition of Vf_m, it is the case that, for all $1 \leq i \leq m$, $\mathsf{Vf}_m((pk, sk), M_i, \sigma_i) = \mathtt{acc}$ if and only if $\mathsf{Vf}_s(pk, M_i, \sigma_i) = \mathtt{acc}$. Thus, if F_m's outputs are valid message-tag pairs under $K = (pk, sk)$, then F_s's outputs are also valid message-signature pairs under pk. Since F_s interacts with its oracle the same number of sessions as F_m does, if F_m uses strictly fewer sessions than the number of output pairs, then so does F_s. Thus, if F_m succeeds, then so does F_s, or

$$\mathbf{Adv}^{\text{omu-sa}}_{\mathcal{BMAC}, \mathsf{F}_m}(k) \leq \mathbf{Adv}^{\text{omu-sa}}_{\mathcal{BS}, \mathsf{F}_s}(k) \ ,$$

which proves the lemma.

Proof (Lemma 8). We prove the lemma via a standard reduction, namely, we assume the existence of an adversary A_m attacking the blindness of $\mathcal{BMAC}$, and then construct an adversary A_s attacking the dishonest-key blindness of $\mathcal{BS}$ so that, if the success probability of the former is non-negligible, then so is that of the latter. The idea is for A_s to first run A_m, and to output the public key contained in the first message of A_m's first user interaction as the public key with which both User_s sessions should be run. The rest of the messages are then relayed faithfully between A_m and the User_s sessions. In A_m's second User_m interaction, the first outgoing message from A_m is simply dropped.

Now we provide more details of how A_s works. We emphasize that A_s operates in the dishonest-key model. On input 1^k, the adversary A_s generates a key pair (pk, sk) via $\mathsf{Kg}(1^k)$, runs $\mathsf{A}_m(1^k, (pk, sk))$, obtains A_m's challenge messages M_0, M_1, and waits until A_m outputs its first outgoing message pk' as part of a User_m session. Then, A_s outputs pk' as the public key for the users along with the same challenge messages M_0, M_1. Adversary A_s relays messages faithfully between User_s and A_m (who is acting in the role of the tagger) for the rest of the interaction. The interaction with the second user is similar: A_s drops the first message from A_m and simply relays following messages to and from its second User_s session. Finally, when given $\sigma = (\sigma_0, \sigma_1)$ or $\mathtt{fail}$, A_s forwards σ to A_m and outputs A_m's guess d as its own.

We first argue that A_s simulates A_m in the same environment as that in Definition 3. Consider the three phases in A_m's attack: starting, interacting with users, and guessing. In the first phase, A_s starts A_m with a legitimate key pair which is indeed what A_m expects. Since, by definition of $\mathcal{BMAC}$, User_s outputs whatever User_m outputs, the tags that A_s gives to A_m in the last phase are also correctly distributed. For the second phase, recall that A_s drops the first

message received from A_m and relays messages between User_s and A_m. Thus, the messages relayed to A_m are exactly what A_m would see in its role as a tagger. Therefore, this phase also follows the correct distribution.

Now suppose that A_m succeeds. We argue that A_s does too. Let pk' be the first outgoing message that A_m outputs to start the session with the "first" user. Let $b \in \{0,1\}$ such that interaction $[\mathsf{A}_s \leftrightarrow \mathsf{User}_s(pk', M_b)]$ starts first. Recall that A_s simulates $\mathsf{User}_m(1^k, \cdot)$ using $\mathsf{User}_s(pk', \cdot)$. This means that the interaction $[\mathsf{A}_m \leftrightarrow \mathsf{User}_m(pk', M_b)]$ also starts first. Since A_s outputs the same answer as A_m, A_s guess correctly whenever A_m does. So we have

$$\mathbf{Adv}^{\text{blind-sa}}_{\mathcal{BMAC},\,\mathsf{A}_m}(k) \leq \mathbf{Adv}^{\text{dk-blind-sa}}_{\mathcal{BS},\,\mathsf{A}_s}(k) \ ,$$

which concludes the proof.

Existence of dishonest-key blind signature schemes. We describe a variant of Chaum's blind signature scheme here. Theorem 9 below states that this scheme is one-more unforgeable and dishonest-key blind. Recall that in Chaum's RSA-based blind signature scheme, the public key is (N, e) and the private key is (N, d) where N is an RSA modulus, e is an RSA encryption exponent, and d is the corresponding RSA decryption exponent. On inputs a public key (N, e) and a message M, the user computes $\overline{M} \leftarrow r^e \cdot H(M) \bmod N$, where r is a random value in $\mathbb{Z}_N^*$ and $H : \{0,1\}^* \to \mathbb{Z}_N^*$ is a public hash function, then submits $\overline{M}$ to the signer. The signer then responds with $\overline{\sigma} \leftarrow \overline{M}^d \bmod N$. Finally, the user computes and outputs $\sigma \leftarrow r^{-1} \cdot \overline{\sigma} \bmod N$. A message-signature pair (M, σ) is valid if and only if $\sigma^e \equiv H(M) \bmod N$. The variant that we are interested in is Chaum's scheme with the additional requirements that e is prime and that $e > N$. The user checks that these requirements hold before starting the protocol, and checks that $\sigma \in \mathbb{Z}_N^*$ and $\sigma^e \equiv H(M) \bmod N$ at the end of the protocol. If any of these checks fail, the User algorithm terminates in a `fail` state. We note that this check can be done in deterministic polynomial time [3].

Theorem 9 [**Security of modified Chaum scheme.**] Let $H : \{0,1\}^* \to \mathbb{Z}_N^*$ be a random oracle, and let $\mathcal{BS}$ be Chaum's blind signature scheme with prime encryption exponent $e > N$. Then, $\mathcal{BS}$ is one-more unforgeable under sequential attacks in the random oracle model assuming that the one-more-RSA-inversion problem is hard. Furthermore, $\mathcal{BS}$ is unconditionally dishonest-key blind under sequential attacks.

Proof (Theorem 9). Bellare et al. proved in [4] that Chaum's scheme is one-more unforgeable in the random oracle model assuming that the one-more-RSA-inversion problem is hard. Their proof does not make additional assumptions about the encryption exponent e. Thus, the same security result holds for our variant of Chaum's scheme.

Now we prove the blindness result. Let A be a dishonest-key blindness adversary. Over the course of the experiment, A's inputs are the incoming messages from the two users and the two resulting signatures. Consider the two worlds

dictated by which message is signed first (i.e. $b = 0$ or $b = 1$) and regard each input of A as a random variable. We argue that each of these random variables has the same distribution in both worlds. We consider them one by one. First, we consider an incoming message $\overline{M}$, which is computed as $r^e \cdot H(M) \bmod N$ where r is a random value in $\mathbb{Z}_N^*$. Since e is prime and $e > N$, we have that $\gcd(e, \phi(N)) = 1$ where $\phi(N)$ is the Euler's totient function. Thus, the map $f : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ defined as $f(x) = x^e \bmod N$ is a permutation on $\mathbb{Z}_N^*$. Consequently, given that r is a random value in $\mathbb{Z}_N^*$, we have that r^e is also a random value in $\mathbb{Z}_N^*$. Thus, so is $\overline{M}$. This is true regardless of the value of b. Therefore, the random variable $\overline{M}$ follows the same distribution in both worlds, namely a uniform distribution over $\mathbb{Z}_N^*$.

Second, we consider a signature σ resulting from A's interaction with a user. At the end of the protocol, the user verified that σ is an element of $\mathbb{Z}_N^*$ such that $\sigma^e \equiv H(M) \bmod N$. Since $f(x)$ is a permutation over $\mathbb{Z}_N^*$, there is only one such element σ. Therefore, σ is uniquely determined by (N, e, M), and in particular does not contain any information about during which session it was created. Thus, $\mathcal{BS}$ is dishonest-key blind.

As a corollary, it follows that blind MAC schemes with state-sharing users that are at the same time one-more unforgeable (omu-sa secure) and blind (blind-sa secure) exist in the random oracle model if the one-more RSA-inversion problem is hard.

6 Parallel Attacks

We note that all our results can be extended to parallel attacks, i.e. attacks where the adversary can interact with signers, taggers or users in an arbitrarily interleaved way. We refer to the full version [1] for notation and security notions under parallel attacks, and simply summarize the results here.

Since any blind MAC scheme that is secure under parallel attacks is also secure under sequential attacks, our impossibility result of Theorem 4 directly implies that secure blind MACs under parallel attacks do not exist either.

In the state-sharing users setting, the result of Theorem 6 easily extends to parallel attacks: if the underlying blind signature scheme is one-more unforgeable and dishonest-key blind under parallel attacks, then the blind MAC scheme of Construction 5 is one-more unforgeable and blind under parallel attacks. Moreover, since the signing protocol in Chaum's scheme only has two moves, security under sequential and parallel attacks are equivalent, and the result of Theorem 9 holds for parallel attacks as well.

7 Future Work

In forthcoming work, we will further explore the notion of dishonest-key blindness for other schemes than the modified Chaum scheme presented in Section 5. The latter relies on random oracles and the one-more RSA-inversion assumption;

we will investigate which other schemes satisfy the stronger notion, and whether a general transformation exists that converts any honest-key blind signature scheme into a dishonest-key blind signature scheme.

As previously stated, the sole purpose of the construction in Section 5 is to demonstrate the existence of blind MAC schemes in the setting in which the users share a common state information. Finding efficient constructions in this setting is left as an open problem. Also, one could investigate the existence of blind MACs in other models, such as a model in which users can collude with a cheating signer, or one in which all users have access to a verification oracle.

8 Acknowledgements

We would like to thank Mihir Bellare and the anonymous reviewers for their valuable suggestions. The first and third author were supported in part by the French RNRT Project Crypto++ and by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT. The third author is a Postdoctoral Fellow of the Research Foundation – Flanders (FWO-Vlaanderen), and was supported in part by the Flemish Government under GOA Mefisto 2006/06 and Ambiorix 2005/11, and by the European Commission through the IST Project PRIME.

References

1. Michel Abdalla, Chanathip Namprempre, and Gregory Neven. On the (im)possibility of blind message authentication codes. Full version of current paper. Available from authors' web pages.
2. Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 136–151, Innsbruck, Austria, May 6–10, 2001. Springer-Verlag, Berlin, Germany.
3. Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. `http://www.cse.iitk.ac.in/users/manindra/primality.ps`, August 2002.
4. Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, 2003.
5. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46, Miami, USA, January 6–8, 2003. Springer-Verlag, Berlin, Germany.
6. Jan Camenisch, Maciej Koprowski, and Bogdan Warinschi. Efficient blind signatures without random oracles. In Carlo Blundo and Stelvio Cimato, editors, *SCN 04: 4th International Conference on Security in Communication Networks*, Lecture Notes in Computer Science, pages 134–148, Amalfi, Italy, September 8–10, 2005. Springer-Verlag, Berlin, Germany.

7. David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – CRYPTO'82*, pages 199–203, Santa Barbara, CA, USA, 1983. Plenum Press, New York, USA.
8. David Chaum. Blind signature system. In David Chaum, editor, *Advances in Cryptology – CRYPTO'83*, page 153, Santa Barbara, CA, USA, 1984. Plenum Press, New York, USA.
9. David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327, Santa Barbara, CA, USA, August 21–25, 1990. Springer-Verlag, Berlin, Germany.
10. Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Josef Pieprzyk, editors, *Advances in Cryptology – AUSCRYPT ' 92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer-Verlag, Berlin, Germany, 1993.
11. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.
12. Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures (Extended abstract). In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 150–164, Santa Barbara, CA, USA, August 17–21, 1997. Springer-Verlag, Berlin, Germany.
13. Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In Bruce Christianson, Bruno Crispo, T. Mark A. Lomas, and Michael Roe, editors, *Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997, Proceedings*, volume 1361 of *Lecture Notes in Computer Science*, pages 25–35. Springer-Verlag, Berlin, Germany, 1998.
14. Benny Pinkas. Fair secure two-party computation. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 87–105, Warsaw, Poland, May 4–8, 2003. Springer-Verlag, Berlin, Germany.
15. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
16. Berry Schoenmakers. Personal Communication, August 2005.
17. Andrew C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Chicago, Illinois, November 3–5, 1982. IEEE Computer Society Press.

A Formal Definition of Dishonest-Key Blindness

The concept of dishonest-key blindness for blind signature schemes is an extension of the classical notion of blindness in which the adversary is allowed to choose the public key used by the user algorithm when trying to break the blindness of the scheme. In particular, in the experiment defining this new notion, no key generation is performed and no key pair is given to the adversary as input to its first phase. Instead, the adversary outputs the public key of its choice along with the challenge messages at the end of its first stage. It is this public key that is given as input to the users during the second phase of the experiment defining dishonest-key blindness.

Definition 10 [**Dishonest-key blindness of a blind signature scheme.**] Let $\mathcal{BS} = (\mathsf{Kg}, \mathsf{User}, \mathsf{Sign}, \mathsf{Vf})$ be a blind signature scheme. Let $k \in \mathbb{N}$, and let A be an adversary. Consider the following experiment.

Experiment $\mathbf{Exp}^{\text{dk-blind-sa}}_{\mathcal{BS},\mathsf{A}}(k)$:
$b \xleftarrow{\$} \{0,1\}$
$((M_0, M_1, pk), St_{\mathsf{A}}) \xleftarrow{\$} \mathsf{A}(\varepsilon, 1^k)$ // *A outputs pk of its choice*
// *both users use pk output by A during the attack*
$(M_{\mathsf{A}}, St_{\mathsf{A}}, \tau_b, St_b) \xleftarrow{\$} [\mathsf{A}(St_{\mathsf{A}}) \leftrightarrow \mathsf{User}((pk, M_b))]$
$(M_{\mathsf{A}}, St_{\mathsf{A}}, \tau_{1-b}, St_{1-b}) \xleftarrow{\$} [\mathsf{A}(St_{\mathsf{A}}) \leftrightarrow \mathsf{User}((pk, M_{1-b}))]$
If $St_0 = \texttt{fail}$ or $St_1 = \texttt{fail}$ then $\tau \leftarrow \texttt{fail}$ else $\tau \leftarrow (\tau_0, \tau_1)$
$d \xleftarrow{\$} \mathsf{A}(\tau, St_{\mathsf{A}})$
If $b = d$ then return 1 else return 0

The dk-blind-sa-advantage of A in breaking $\mathcal{BS}$ is defined as

$$\mathbf{Adv}^{\text{dk-blind-sa}}_{\mathcal{BS},\mathsf{A}}(k) \;=\; 2 \cdot \Pr\left[\,\mathbf{Exp}^{\text{dk-blind-sa}}_{\mathcal{BS},\mathsf{A}}(k) = 1\,\right] - 1\;,$$

and $\mathcal{BS}$ is said to be *dishonest-key blind under sequential attacks or* dk-blind-sa-*secure* if $\mathbf{Adv}^{\text{dk-blind-sa}}_{\mathcal{BS},\mathsf{A}}(k)$ is a negligible function in the security parameter k for all adversaries A with time complexity polynomial in k. ▌