A Study of Blind Message Authentication Codes

Article in IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences - January 2007
DOI: 10.1093/ietfec/e90-a.1.75 - Source: DBLP

3 authors:

Chanathip Namprempre
Thammasat University
35 PUBLICATIONS 2,924 CITATIONS

Michel Abdalla
Ecole Normale Supérieure de Paris
104 PUBLICATIONS 4,217 CITATIONS

Some of the authors of this publication are also working on these related projects:

Identity Mixer View project
UC-Security View project
SUMMARY 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*. 

key words: 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], [12] to protect voters’ privacy. Since their first introduction in 1982, blind signatures have become a well-studied primitive with formal security notions [11], [15], practical schemes realizing these notions under various assumptions [2], [5], [6], [8], [15], and a theoretical construction based on the existence of trapdoor one-way permutations [11].

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 \( \tau \) 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 \((M, \tau)\), 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

Copyright © 2007 The Institute of Electronics, Information and Communication Engineers
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 and adds the coin’s identifying string to the database. Since it transfers the correct amount to the merchant’s account, in a database whether the coin is being double-spent. If not, the bank checks the validity of the signature again, and looks up in a

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 4.1, we show that the natural symmetric analogs of the one-more unforgeability [15] and blindness [11] 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 Sect. 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].

### 2. Blind Signatures

**Notation.** We let $N = \{1, 2, 3, \ldots\}$ denote the set of natural numbers. If $k \in N$, then $1^k$ is the string of $k$ ones. The empty string is denoted $\varepsilon$. 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 $A(x_1, x_2, \ldots : O_1, O_2, \ldots)$ means that $A$ has inputs $x_1, x_2, \ldots$ and access to oracles $O_1, O_2, \ldots$. Also $y \leftarrow A(x_1, x_2, \ldots : O_1, O_2, \ldots)$ means that we run the randomized algorithm $A$ on inputs $x_1, x_2, \ldots$ and with access to oracles $O_1, O_2, \ldots$, and let $y$ denote the output obtained.

An **interactive algorithm** is a stateful algorithm that on input an incoming message $M_\text{in}$ (this is $\varepsilon$ if the party is initiating the protocol) and state information $St$ outputs an outgoing message $M_\text{out}$ and updated state $St'$. For an interactive algorithm $A$ having access to oracles $O_1, O_2, \ldots$, this is written as $(M_\text{out}, St') \leftarrow A(M_\text{in}, St : O_1, O_2, \ldots)$. 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 $(M_A, S_A, M_B, S_B) \leftarrow [A(S_A) \leftrightarrow B(S_B)]$ to denote the final outgoing messages and states after an interaction between $A$ and $B$ when run on initial states $S_A$ and $S_B$, respectively.

**Syntax of Blind Signatures.** We repeat the definition of blind signatures as proposed by Juels et al. [11]. A blind signature scheme $\mathcal{B}$ is a tuple of four polynomial-time algorithms $(Kg, User, Sign, Vf)$ 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 $User$ on an initial state consisting of a public key $pk$ and a message $M \in \{0, 1\}^*$, and lets it interact with $Sign$ that is run by the signer on initial state a secret key $sk$. At the end of the protocol, $User$ either enters the halt state and outputs a signature $\sigma$ as its last outgoing message, or enters the fail state to indicate failure. $Sign$ 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 $\sigma$ 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 $S_{User} = \text{halt}$ and $Vf(pk, M, \sigma) = \text{acc}$ when $(pk, sk) \leftarrow Kg(1^k)$ and $(M_{Sign}, S_{Sign}, \sigma, S_{User}) \leftarrow [Sign(sk) \leftrightarrow User((pk, M))]$ with probability 1.

**Unforgeability of Blind Signatures.** The security of a blind signature scheme is twofold: on the one hand, a user should not 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).

We use the definition of one-more unforgeability of Pointcheval and Stern [15]. Let $\mathcal{B}$ 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 $(pk, sk) \leftarrow Kg(1^k)$, and runs $A$ on input $(1^k, pk)$. The adversary has access to a signing oracle that runs the $Sign(sk, \cdot)$ 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 Sect. 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 $Vf(pk, M_i, \sigma_i) = \text{acc}$ for all $1 \leq i \leq m$, all $M_i$ are different and $m > n$.

The advantage function $Adv_{\mathcal{B}, A}^{\text{omua}}(k)$ is defined as $A$’s probability of winning the above game, and $\mathcal{B}$ is said to be one-more unforgeable under sequential attacks (omu-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. [11]. 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 $User(pk, M_0)$ and the second is with $User(pk, M_1)$; if $b = 1$, then $A$ first interacts with $User(pk, M_1)$ and then with $User(pk, M_0)$. If $A$ accepts in both sessions, then $A$ is additionally given the resulting signatures $\sigma_0, \sigma_1$ for messages $M_0, M_1$, outputs its guess $d$, and wins the game if $b = d$. The advantage $Adv_{\mathcal{B}, A}^{\text{sec}}(k)$ is defined as $2p - 1$, where $p$ is the probability that $A$ wins this game. The scheme $\mathcal{B}$ 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 of the paper [1] for formal descriptions of the security experiments.

### 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 3.1:** [Syntax of a blind MAC scheme] A blind MAC scheme $\mathcal{B}$ is a tuple of four polynomial-time algorithms $(Kg, User, Tag, Vf)$ where

- the randomized key generation algorithm $Kg$, on input $1^k$ 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 $K^\dagger$. At the end of the protocol,

\footnote{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 User algorithm either enters the halt state and outputs a MAC value \( r \) 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 \( r \) as input, and outputs acc or rej to indicate acceptance or rejection of the MAC value, respectively.

Let \( M \) and \( Kg \) denote the message and the secret key of the MAC algorithm respectively. In the blindness game, the experiment chooses a random bit \( k \) and generates a fresh key \( Kg(1^k) \) and (\( M_{Tag}, St_{Tag}, r, St_{User} \)) \( \sim \) \([\text{Tag}(K) \leftrightarrow \text{User}((1^k, M)])\).

Security of Blind MACs. Analogously to blind signatures, the security of a blind MAC scheme consists of an unforgeability and a blindness requirement. For unforgeability, the experiment generates a fresh key \( Kg(1^k) \), 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 and 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.

Definition 3.2: [Unforgeability of a blind MAC scheme.] Let \( \text{BMAC} = (Kg, User, Tag, Vf) \) 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 \( \text{Exp}_{\text{omu-sa}}^{\text{BMAC}, A}(k) \):

\[
K \leftarrow Kg(1^k); \ n \leftarrow 0
\]

\[
(M_1, \tau_1), \ldots, (M_m, \tau_m) \leftarrow A(1^k : \text{TAG}(i))
\]

If \( Vf(K, M_i, \tau_i) = \text{acc} \) for all \( 1 \leq i \leq m \) and \( m > n \) and \( M_i \neq M_j \) for all \( 1 \leq i < j \leq m \), then return 1 else return 0,

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

Oracle \( \text{TAG}(M_in) \):

If \( M_in = \perp \) then \( St_{Tag} \leftarrow K ; M_out \leftarrow \perp \)
else \( (M_in, St_{Tag}) \leftarrow \text{TAG}(M_in, St_{Tag}[s]) \)
If \( St_{Tag} = \text{halt} \) then \( n \leftarrow n + 1 \)
Return \( M_out \)

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

\[
\text{Adv}_{\text{omu-sa}}^{\text{BMAC}, A}(k) = \Pr[\text{Exp}_{\text{omu-sa}}^{\text{BMAC}, A}(k) = 1],
\]

and \( \text{BMAC} \) is said to be one-more unforgeable under sequential attacks (omu-sa-secure) if the advantage \( \text{Adv}_{\text{omu-sa}}^{\text{BMAC}, A}(k) \) 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. In particular, we demonstrate a universal blindness adversary \( A \) and a universal forger \( F \) so that for any candidate scheme, one of them always wins with non-negligible probability.

Theorem 4.1: [Secure blind MAC schemes do not exist.] Let \( \text{BMAC} \) be a blind MAC scheme. Either \( \text{BMAC} \) is one-more forgeable under sequential attacks, or it is not blind under sequential attacks.

Proof of Theorem 4.1: We define an adversary \( A \) breaking the blindness of \( \text{BMAC} \) and an adversary \( F \) breaking the one-more unforgeability of \( \text{BMAC} \), both under sequential attacks, so that
\[ \text{Adv}^{\text{blind-sa}}_{\text{BMAC}, A}(k) + \text{Adv}^{\text{omu-sa}}_{\text{BMAC}, F}(k) = 1, \]

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. On initial state \((1^k, K)\), \( A \) generates a second key \( K' \leftarrow \text{Kg}(1^k) \) 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 \( \text{Tag}(K) \), and with the second by running \( \text{Tag}(K') \). Since both \( K \) and \( K' \) are keys generated by the \( \text{Kg} \) algorithm, the correctness requirement for \( \text{BMAC} \) implies that neither of the user sessions fails, and hence that \( A \) gets back tags \((\tau_0, \tau_1)\). If \( \forall i(K, M_0, \tau_0) = \text{acc} \), 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 \( K' \leftarrow \text{Kg}(1^k) \). It simulates an interaction \((M, \text{St}_{\text{Tag}}, \tau, \text{St}_{\text{User}}) \leftarrow [\text{Tag}(K') \leftrightarrow \text{User}((1^k, M_0))] \) 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 3.3,

\[
\text{Adv}^{\text{blind-sa}}_{\text{BMAC}, A}(k) = 2 \cdot \text{Pr}[\text{Exp}^{\text{blind-sa}}_{\text{BMAC}, A}(k) = 1 | b = 1] - 1
\]

\[
= \text{Pr}[\text{Exp}^{\text{blind-sa}}_{\text{BMAC}, A}(k) = 1 | b = 1] + \text{Pr}[\text{Exp}^{\text{blind-sa}}_{\text{BMAC}, A}(k) = 1 | b = 0] - 1
\]

\[
= \text{Pr}[\text{Exp}^{\text{blind-sa}}_{\text{BMAC}, A}(k) = 1 | b = 1] - 1 - \text{Adv}^{\text{omu-sa}}_{\text{BMAC}, F}(k)
\]

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 \( \text{BMAC} \); 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. The probability on the third line is one minus the probability that a tag \( \tau_0 \) obtained from an interaction \([\text{Tag}(K') \leftrightarrow \text{User}((1^k, M_0))]\) 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.

5. Blind MACs for State-Sharing Users

The attack in Sect. 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
The dishonest-key blindness notion that we propose gives way that allows him to break the blindness of the scheme. where the signer creates a public key in a special, malicious sary is given a key pair generated properly through the key blindness notion for blind signatures assumes that the adver- is needed to prove its security. Recall that the standard describe here the concept of dishonest-key blindness, which describing the security of our blind MAC construction, we briefly D
BMAC
input string CS
User
sk
User
M
k
Vf
Adv
IEICE TRANS. FUNDAMENTALS, VOL.E90–A, NO.1 JANUARY 2007

A CONSTRUCTION BASED ON BLIND SIGNATURES. The main idea for the 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 5.1: [A blind MAC scheme for state-sharing users.] Let \( BS = (K_g, User_m, Sign, V_f_i) \) be a blind signature scheme. We associate it to a blind MAC scheme \( BMAC = (K_g, User_m, Tag, V_f_m) \):

- On input \( 1^k \), the key generation algorithm \( K_g_m \) runs \( K_g(1^k) \) 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 \( CS \), the algorithm \( User_m \) first receives \( pk \) from \( Tag \). If \( CS = e \), then \( User_m \) sets \( CS \leftarrow pk \). Otherwise, it sends \( pk \leftarrow CS \) and runs \( User_m \) on the initial state \( (pk, M) \) interacting with \( Tag \) until the interaction completes. It sets its state and output to those of \( User_m \).
- On input a key \( K \), a message \( M \), and a MAC value \( \tau \), the algorithm \( V_f_m \) parses \( K \) as \((pk, sk)\), and returns \( V_f,(pk, M, \tau) \).

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 \( K_g \), nor does the adversary need to know the corresponding secret key. The adversary gets the input \( 1^k \), and outputs challenge messages \( M_0, M_1 \) and 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 \( b \).

We construct a secure blind MAC in the state-sharing model based on this stronger security requirement of the underlying blind signature scheme. Let \( Adv_{BS}^{dk-blind-sa}(k) \) be the advantage of an adversary \( A \) in winning the above game against \( BS \) in a sequential attack. We say that \( BS \) is dk-blind-sa-secure if this advantage is a negligible function in \( k \) for all polynomial-time algorithms \( A \). See the full ver- sion of this paper [1] for a formal definition of dishonest-key blindness.

Security. The following theorem states that, if the under- lying blind signature scheme is one-more unforgeable and dishonest-key blind, then the resulting blind MAC scheme is secure. The proof follows directly from the lemmas below. For brevity, we provide only their proof sketches here.

Theorem 5.2: If a blind signature scheme \( BS \) is one-more unforgeable and dishonest-key blind, then the resulting blind MAC scheme is also one-more unforgeable and blind under sequential attacks.

Lemma 5.3: If \( BS \) is omu-sa secure, then \( BMAC \) is also omu-sa secure.

Lemma 5.4: If \( BS \) is dk-blind-sa secure, then \( BMAC \) is blind-sa-secure.

Sketch of Proof of Lemma 5.3: Suppose that a forger \( F_m \) attacking \( BMAC \) exists. We construct a forger \( F_s \) to attack \( BS \) 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 \( Tag() \). Since the only difference between a tagger-user interaction in \( BMAC \) and a signer-user interaction in \( BS \) 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.

Sketch of Proof of Lemma 5.4: Suppose that an adversary \( A_m \) attacking the blindness of \( BMAC \) exists. We construct \( A_s \) attacking the dishonest-key blindness of \( BS \) 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 first user inter- action is simulated by relaying messages between \( A_m \) and the \( User \) session. For the second user interaction, \( A_s \) ignores the public key sent in the first move by \( A_m \), and again relays messages between \( A_m \) and \( User \). Eventually, \( A_s \) outputs the same bit as \( A_m \).

Existence of dishonest-key blind signature schemes. We describe a variant of Chaum’s blind signature scheme here. Theorem 5.5 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 $(N,e)$ and a message $M$, the user computes $\overline{M} \leftarrow r^e \cdot H(M) \mod N$, where $r$ is a random value in $\mathbb{Z}_N^*$ and $H : \{0,1\}^* \rightarrow \mathbb{Z}_N^*$ is a public hash function, then submits $\overline{M}$ to the signer who then responds with $\overline{\sigma} \leftarrow \overline{M}^d \mod N$. Finally, the user computes and outputs $\sigma \leftarrow r^{-1} \cdot \overline{\sigma} \mod N$. A pair $(M, \sigma)$ is valid if and only if $\sigma^e \equiv H(M) \mod N$. The variant that we are interested in is Chaum’s scheme with the additional requirements that $e > N$ and is prime. The user verifies these responses before starting the protocol, and checks that $\sigma \in \mathbb{Z}_N^*$ and $\sigma^e \equiv H(M) \mod N$ at the end of the protocol. If any of these checks fail, User terminates in a fail state. We note that this check can be done in deterministic polynomial time [3].

**Theorem 5.5:** [Security of modified Chaum scheme.] Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_N^*$ 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 of Theorem 5.5:** 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 $r^e \cdot H(M) \mod N$ 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 $f : \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$ defined as $f(x) = x^e \mod N$ 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 $\sigma$ resulting from $A$’s interaction with a user. When the protocol ends, the user verified that $\sigma$ is in $\mathbb{Z}_N^*$ such that $\sigma^e \equiv H(M) \mod N$. Since $f(x)$ is a permutation over $\mathbb{Z}_N^*$, there is only one such element $\sigma'$. Therefore, $\sigma$ 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. We note that independently of our work, Okamoto [13] recently introduced the same notion of dishonest-key blindness and proposed a scheme that satisfies it in the standard model under the so-called 2SDH assumption.

On the equivalence with blind signatures. While at first sight one may suspect that blind MACs with state-sharing users imply blind signatures, we have not been able to prove this. The natural way to construct a blind signature scheme would be to use the output of the blind MAC key generation algorithm as the secret signing key, and to use the shared state information as the public verification key. It is not clear however how one can verify signatures in this scheme. While in the particular case of Construction 5.1 the shared state information can be used as a verification key, this is not necessarily true for any blind MAC scheme in general. Therefore, the equivalence of blind signatures and blind MACs with state-sharing users remains an open problem.

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 formal 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 a fortiori also secure under sequential attacks, the impossibility result of Theorem 4.1 for sequential attacks directly implies the non-existence of blind MACs under parallel attacks. For blind signature (or MAC) schemes with two-move signing (or tagging) protocols, like Chaum’s, security under sequential and parallel attacks are obviously equivalent. Therefore, when instantiated with Chaum’s blind signature scheme, Construction 5.1 immediately proves the existence of blind MACs with state-sharing users under parallel attacks as well.

Acknowledgments

We thank Mihir Bellare and the reviewers for their valuable suggestions. The first author was supported by Thailand Research Fund. The second and third authors were supported in part by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT. The second 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


Chanathip Namprempre received the S.B. and M.Eng. in electrical engineering and computer science from MIT, and the Ph.D. in computer science from the University of California, San Diego. She is currently an assistant professor in the Faculty of Engineering, Thammasat University, Thailand. Her research interests are design and analysis of cryptographic protocols.

Gregory Neven received his Master’s degree in Computer Science Engineering from Katholieke Universiteit Leuven (K.U. Leuven) in 2000. In 2004 he obtained his Ph.D. in Applied Sciences from the same university, supported by research assistantship from the Flemish Fund for Scientific Research. During his Ph.D., he spent fifteen months at University of California San Diego (UCSD) working with Mihir Bellare on various topics in provable security. He is currently a postdoctoral fellow of the Flemish Fund for Scientific Research at K.U. Leuven and Ecole Normale Supérieure (ENS).

Michel Abdalla is currently a researcher with the Centre National de la Recherche Scientifique (CNRS), France, and a member of the cryptography group at the Ecole Normale Supérieure (ENS), France. He holds a B.Sc. degree in Electronics Engineering and a M.Sc. degree in Electric Engineering from the Universidade Federal do Rio de Janeiro (UFRJ), Brazil, and a Ph.D. degree in Computer Science from the University of California, San Diego (UCSD), USA. He is mainly interested in the design of efficient and provably-secure cryptographic protocols.