Security of Blind Signatures Revisited

Dominique Schröder*[1] and Dominique Unruh[2]

[1] University of Maryland, USA
[2] University of Tartu, Estonia

Abstract. We revisit the definition of unforgeability of blind signatures as proposed by Pointcheval and Stern (Journal of Cryptology 2000). Surprisingly, we show that this established definition falls short in two ways of what one would intuitively expect from a secure blind signature scheme: It is not excluded that an adversary submits the same message m twice for signing, and then produces a signature for $m' \neq m$. The reason is that the forger only succeeds if *all* messages are distinct. Moreover, it is not excluded that an adversary performs k signing queries and produces signatures on $k + 1$ messages as long as *each* of these signatures does not pass verification with probability 1.

Finally, we propose a new definition, honest-user unforgeability, that covers these attacks. We give a simple and efficient transformation that transforms any unforgeable blind signature scheme (with deterministic verification) into an honest-user unforgeable one.

1 Introduction

Blind signature schemes have been suggested by Chaum [12,13]. Roughly speaking, this widely-studied primitive allows a signer to interactively issue signatures for a user such that the signer learns nothing about the message being signed (*blindness*) while the user cannot compute any additional signature without the help of the signer (*unforgeability*). Typical applications of blind signatures include e-cash, where a bank signs coins withdrawn by users, and e-voting, where an authority signs public keys that voters later use to cast their votes. Another application of blind signature schemes are anonymous credentials, where the issuing authority blindly signs a key [9,10]. Very recently, Microsoft introduced a new technology called U-Prove to "overcome the long standing dilemma between identity assurance and privacy" [6,29]. Their technology uses as a central building block blind signatures [6,8].

There are two main security requirements for blind signature schemes. First, the scheme should be blind. That is, a malicious signer should not be able to link the final signatures output by the user to the individual interactions with the user. In other words, the signer cannot tell which session of the signing protocol corresponds to which message. Second, the scheme should be unforgeable. That is, an adversary, even if he can impersonate the user and interact freely with the signer, should not be able to produce signatures on messages except for those

* Supported in part by a DAAD postdoctoral fellowship.

that the signer signed. It is the notion of unforgeability we are concerned with in this paper.

A formal definition of the unforgeability of blind signatures schemes (or generally interactive signature schemes) has been proposed by [25]. Roughly, their definition states that an adversary that interacts k times with the adversary cannot produce valid signatures on more than k different messages.[3] At this point, one may wonder why the definition of unforgeability does not just require that the adversary cannot output a signature for m unless there was an interaction with the signer in which m was queried. The reason is that in general, it is not well-defined which message is queried in a given interaction. The message is not sent in clear, and it might be even information-theoretically impossible to tell from an interaction which message is being signed.[4] Thus, in order to be able to tell which message is signed in a given interaction, we would have to add some kind of extractability to the security definition; this would be an additional requirement on the protocols and make them more complex.

Insecurity of unforgeable blind signatures schemes. Unfortunately, however, the definition of unforgeability might not cover all cases in which one would intuitively expect unforgeability to be sufficient. We illustrate this by the following toy protocol:

Consider the setting of an online video store such as Netflix. In our setting, we assume that the store is implemented via two entities, the content provider and the reseller. We assume that the contract between client and reseller is a flatrate that allows the client to download a fixed number of movies. For privacy reasons, we do not wish the reseller to know which movies the client actually watches. On the other hand, we wish to ensure that underage clients can only download movies suitable for their age. To achieve this, we introduce another (trusted) entity, the parental control server whose job it is to work as a proxy between reseller and client and to ensure that the client only obtains appropriate movies. Then, to download a movie X, the client first sends her name and X to the parental control server. If X is appropriate for the client, the parental control server then runs a blind signature scheme with the reseller to obtain a signature

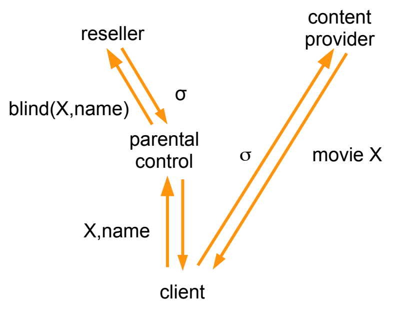


Fig. 1. Setting of an online video store.

[3] There is also a variant called strong unforgeability which requires that the adversary cannot produce more than k different message/signature pairs. In particular, this means that the adversary wins even if he produces additional signatures for an already signed message. Since most known blind signature schemes (e.g., [20,15,3,26,19,18]) do not satisfy strong unforgeability, in this work we focus on the weaker notion.

[4] This might be the case when signing a message m is implemented by signing an information-theoretically hiding commitment on m.

σ on (X, name) (the blind signature is used to protect the privacy of the client, there is no need for the reseller to know which movies the client watches). Then σ is sent to the client, and the client uses σ to download X from the content provider. (We assume that all communication is suitably authenticated.)

At a first glance, it seems that this protocol is secure. In particular, the client will not be able to download a movie that is not approved by the parental control server. It turns out, however, that the client can cheat the parental control server: Assume the client twice requests a signature on some harmless movie X. He will then obtain two signatures σ_1, σ_2 on X from the parental control server. Then, given σ_1 and σ_2, the client might be able to compute a signature on an adult movie Y that has not been approved by the parental control server.

It seems that unforgeability should forbid the possibility of such an attack. But it does not. From the point of view of the signer, two signing queries have been performed, and finally signatures on two different messages X and Y have been produced. This does not violate the definition of unforgeability. In fact, we show in Section 4.2 that blind signature schemes exist that allow such attacks but that are still unforgeable.

What went wrong? The definition of unforgeability covers *only partially* the case that the user of the scheme is honest. It only ensures that the number of signed messages is not greater than the number of interactions with the signer. Only considering the number of messages but not their content is fine from the signer's point of view who is not allowed to know the messages anyway. It is not, however, fine from the user's point of view. If the user signs some messages $m_1, \ldots, m_k$ (by interacting with the signer), he expects that no signature on some different message m' can be computed from his signatures. We believe that settings in which the user is honest are natural, and that the definition of unforgeability should cover this case. We thus propose a new *game-based* definition, honest-user unforgeability, which is a strengthening of unforgeability. Alternatively, one could also define an ideal functionality (see [14,4]) that covers these attacks, but schemes that achieve such strong security properties are usually less efficient.

Definition 1 (Honest-user unforgeability – informal). *If an adversary performs k direct interactions with the signer, and requests signatures for the message $m_1, \ldots, m_n$ from the user (which produces these signatures by interacting with the signer), then the adversary cannot produce signatures for pairwise distinct messages $m_1^*, \ldots, m_{k+1}^*$ with $\{m_1^*, \ldots, m_{k+1}^*\} \cap \{m_1, \ldots, m_n\} = \varnothing$.*

Notice that this definition also covers the hybrid case in which the adversary interacts with an honest user and the signer simultaneously. Alternatively, one could also require that security in each of the setting individually: Security when there is no honest user (that is, the normal definition of unforgeability), and security when the adversary may not query the signer directly (we call this $\mathcal{S}+\mathcal{U}$-unforgeability). We show in the full version of this paper [28] that requiring these variants of security individually leads to a strictly weaker security notion. Notice that $\mathcal{S}+\mathcal{U}$-unforgeability would be sufficient to solve the problem in our video store example. It seems, however, restrictive to assume that in all protocols, there will always be only either queries from honest users or only from dishonest users but never from both in the same execution.

Achieving honest-user unforgeability. We show that any unforgeable blind signature scheme can be converted into an honest-user unforgeable blind signature scheme. The transformation is very simple and efficient: Instead of signing a message m, in the transformed scheme the user signs the message (m, r) where r is some randomness. Furthermore, we show that if a scheme is already strongly unforgeable, then it is strongly honest-user unforgeable (as long as the original scheme is *randomized* which holds for most signature schemes).

Insecurity with probabilistic verification. Most interactive and non-interactive signature schemes have a deterministic verification algorithm. In general, however, having a deterministic verification is not a necessity. Yet, when we allow a probabilistic verification algorithm (and this is usually not excluded), both the definition of unforgeability as well as the definition of honest-user unforgeability are subject to an attack: Consider again our video store example. Let λ denote the security parameter. Fix a polynomial $p = p(\lambda) > \lambda$. Assume that the parental control server and the client are malicious and collude. The parental control server interacts with the reseller λ times, and produces p "half-signatures" on movie names $X_1, \ldots, X_p$. Here, a half-signature means a signature that passes verification with probability $\frac{1}{2}$. Then the client can download the movies $X_1, \ldots, X_n$ from the content provider. (If in some download request, a half-signature does not pass verification, the client just retries his request.) Thus the client got p movies, even if his flatrate only allows for downloading λ movies.

Can this happen? It seems that unforgeability would exclude this because $p > \lambda$ signatures were produced using λ queries to the signer. In the definition of unforgeability, however, the adversary succeeds if it outputs $p > \lambda$ signatures such that *all* signatures pass verification. However, the signatures that are produced are half-signatures: That is, the probability that all $p > \lambda$ signatures pass the verification simultaneously is negligible! Thus, producing more than λ half-signatures using λ queries would not be considered an attack by the definition of unforgeability. In Section 5, we show that blind signature schemes exist that allow such attacks but that satisfy the definition of unforgeability. The same applies to honest-user unforgeability as described so far; we thus need to augment the definition further.

There are two solutions to this problem. One is to explicitly require that the verification algorithm is deterministic. Since most schemes have deterministic verification, this is not a strong restriction. To cover the case of probabilistic verification, we propose an augmented definition of honest-user unforgeability in Section 5: This definition considers a list of signatures as a successful forgery if each of them would pass verification with noticeable probability (roughly speaking).

We do not propose a generic transformation that makes schemes with probabilistic verification secure according to our definition. Yet, since most schemes have a deterministic verification anyway; these schemes will automatically satisfy our augmented definition.

Related work. Many blind signature schemes have been proposed in the literature, these schemes differ in their round complexity, their underlying computational assumptions, and the model in which the proof of security is given. For example, some schemes rely on the random oracle heuristic [25,2,5,7,4], some constructions are secure in the standard model [11,24,21,23,17,3,27] ([17,3] assume the existence of a common reference string), and some constructions are based on general assumptions [22,14,20,18,27]. Only a few works consider the security of blind signatures [22,25,15] or their round complexity [16].

Notations. Before presenting our results we briefly recall some basic definitions. In what follows we denote by $\lambda \in \mathbb{N}$ the security parameter. Informally, we say that a function is *negligible* if it vanishes faster than the inverse of any polynomial. We call a function non-negligible if it is not negligible. If S is a set, then $x \overset{\$}{\leftarrow} S$ indicates that x is chosen uniformly at random over S (which in particular assumes that S can be sampled efficiently).

2 Blind signatures

To define blind signatures formally we introduce the following notation for interactive executions between algorithms $\mathcal{X}$ and $\mathcal{Y}$. By $(a, b) \leftarrow \langle \mathcal{X}(x), \mathcal{Y}(y) \rangle$ we denote the joint execution of $\mathcal{X}$ and $\mathcal{Y}$, where x is the private input of $\mathcal{X}$ and y defines the private input of $\mathcal{Y}$. The private output of $\mathcal{X}$ equals a and the private output of $\mathcal{Y}$ is b. We write $\mathcal{Y}^{\langle \mathcal{X}(x), \cdot \rangle^{\infty}}(y)$ if $\mathcal{Y}$ can invoke an unbounded number of executions of the interactive protocol with $\mathcal{X}$ in arbitrarily interleaved order. Accordingly, $\mathcal{X}^{\langle \cdot, \mathcal{Y}(y_0) \rangle^1, \langle \cdot, \mathcal{Y}(y_1) \rangle^1}(x)$ can invoke arbitrarily ordered executions with $\mathcal{Y}(y_0)$ and $\mathcal{Y}(y_1)$, but interact with each algorithm only once.

The invoking oracle machine does not see the private output of the invoked machine. In the above definition this means that $\mathcal{Y}$ does not learn a and $\mathcal{X}$ does not learn b_0 (resp. b_1).

Definition 2 (Interactive signature scheme). *We define an interactive signature scheme as a tuple of efficient*[5] *algorithms* $\mathsf{BS} = (\mathsf{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf})$ *(the key-generation algorithm* KG*, the signer* $\mathcal{S}$*, the user* $\mathcal{U}$*, and the verification algorithm* Vf*) where*

Key Generation. $\mathsf{KG}(1^\lambda)$ *for parameter* λ *generates a key pair* (sk, pk)*.*

[5] More precisely, KG and Vf run in polynomial-time in the total length of their inputs. The total running time of $\mathcal{S}$ is polynomial in the total length of its input (sk) plus the total length of its incoming messages. The total running time of $\mathcal{U}$ is polynomial in the total length of its input (pk, m). (But the running time of $\mathcal{U}$ may not depend on its incoming messages.) The asymmetry between the running time of $\mathcal{S}$ and $\mathcal{U}$ is necessary to ensure that (a) an interaction between $\mathcal{U}$ and $\mathcal{S}$ always runs in polynomial-time, and (b) that the running-time of $\mathcal{S}$ may depend on the length of the message m that only $\mathcal{U}$ has in its input.

Signature Issuing. *The execution of algorithm* $\mathcal{S}(sk)$ *and algorithm* $\mathcal{U}(pk, m)$ *for message* $m \in \{0,1\}^*$ *generates an output* σ *of the user (and some possibly empty output out for the signer.),* $(out, \sigma) \leftarrow \langle \mathcal{S}(sk), \mathcal{U}(pk, m) \rangle$.

Verification. $\mathsf{Vf}(pk, m, \sigma)$ *outputs a bit.*

It is assumed that the scheme is complete*, i.e., for any function* f*, with overwhelming probability in* $\lambda \in \mathbb{N}$ *the following holds: when executing* $(sk, pk) \leftarrow \mathsf{KG}(1^\lambda)$*, setting* $m := f(\lambda, pk, sk)$*, and letting* σ *be the output by* $\mathcal{U}$ *in the joint execution of* $\mathcal{S}(sk)$ *and* $\mathcal{U}(pk, m)$*, then we have* $\mathsf{Vf}(pk, m, \sigma) = 1$.

3 Security of blind signatures

Security of blind signature schemes is defined by unforgeability and blindness [22,25].

Unforgeability. An adversary $\mathcal{U}^*$ against unforgeability tries to generate $k+1$ valid message/signatures pairs with different messages after at most k completed interactions with the honest signer, where the number of executions is adaptively determined by $\mathcal{U}^*$ during the attack. To identify completed sessions we assume that the honest signer returns a special symbol ok when having sent the final protocol message in order to indicate a completed execution (from its point of view). We remark that this output is "atomically" connected to the final transmission to the user.

Definition 3 (Unforgeability). *An interactive signature scheme* $\mathsf{BS} = (\mathsf{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf})$ *is called* unforgeable *if for any efficient algorithm* $\mathcal{A}$ *(the malicious user) the probability that experiment* $\mathsf{Unforge}_{\mathcal{A}}^{\mathsf{BS}}(\lambda)$ *evaluates to* 1 *is negligible (as a function of* λ*) where*

```
Experiment $\mathsf{Unforge}_{\mathcal{A}}^{\mathsf{BS}}(\lambda)$
    $(sk, pk) \leftarrow \mathsf{KG}(1^\lambda)$
    $((m_1^*, \sigma_1^*), \ldots, (m_{k+1}^*, \sigma_{k+1}^*)) \leftarrow \mathcal{A}^{\langle \mathcal{S}(sk), \cdot \rangle^\infty}(pk)$
    Return 1 iff
        $m_i^* \neq m_j^*$ for $i, j$ with $i \neq j$, and
        $\mathsf{Vf}(pk, m_i^*, \sigma_i^*) = 1$ for all $i$, and
        $\mathcal{S}$ has returned $\mathsf{ok}$ in at most $k$ interactions.
```

An interactive signature scheme is *strongly unforgeable* if the condition "$m_i^* \neq m_j^*$ for i, j with $i \neq j$" in the above definition is substituted by "$(m_i^*, \sigma_i^*) \neq (m_j^*, \sigma_j^*)$ for i, j with $i \neq j$".

Observe that the adversary $\mathcal{A}$ does not learn the private output *out* of the signer $\mathcal{S}(sk)$. We assume schemes in which it can be efficiently determined from the interaction between signer and adversary whether the signer outputs ok. If this is not the case, we need to augment the definition and explicitly give the adversary access to the output *out* since *out* might leak information that the adversary could use to produce forgeries.

Blindness. The blindness condition says that it should be infeasible for a malicious signer $\mathcal{S}^*$ to decide which of two messages m_0 and m_1 has been signed first in two executions with an honest user $\mathcal{U}$. This condition must hold, even if $\mathcal{S}^*$ is allowed to choose the public key maliciously [1]. If one of these executions has returned $\perp$ then the signer is not informed about the other signature either.

Definition 4 (Blindness). *A blind signature scheme* $\mathsf{BS} = (\mathsf{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf})$ *is called* blind *if for any efficient algorithm* $\mathcal{S}^*$ *(working in modes* find*,* issue *and* guess*) the probability that the following experiment* $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{S}^*}(\lambda)$ *evaluates to* 1 *is negligibly close to* 1/2*, where*

```
Experiment $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{S}^*}(\lambda)$
    $(pk, m_0, m_1, st_{\mathsf{find}}) \leftarrow \mathcal{S}^*(\mathsf{find}, 1^\lambda)$
    $b \xleftarrow{\$} \{0,1\}$
    $st_{\mathsf{issue}} \leftarrow \mathcal{S}^{*\langle \cdot, \mathcal{U}(pk,m_b)\rangle^1, \langle \cdot, \mathcal{U}(pk,m_{1-b})\rangle^1}(\mathsf{issue}, st_{\mathsf{find}})$
        and let $\sigma_b, \sigma_{1-b}$ denote the (possibly undefined) local outputs
        of $\mathcal{U}(pk, m_b)$ resp. $\mathcal{U}(pk, m_{1-b})$.
    set $(\sigma_0, \sigma_1) = (\perp, \perp)$ if $\sigma_0 = \perp$ or $\sigma_1 = \perp$
    $b^* \leftarrow \mathcal{S}^*(\mathsf{guess}, \sigma_0, \sigma_1, st_{\mathsf{issue}})$
    return 1 iff $b = b^*$.
```

4 Honest-user unforgeability

In this section we introduce a stronger notion of unforgeability that we call *honest-user unforgeability*. In the traditional definition of unforgeability due to [22,25], the adversary fulfills the role of the user. This means that the attacker may choose all messages that are exchanged during the signature issue protocol at will. In particular, the attacker may sample random message *without* fixing a specific message and a certain randomness for the user algorithm. Even if the adversary runs the honest user algorithm, due to the blindness, it is impossible to tell which message has been used. Thus, from a definitional perspective, one has to count the number of executions and produced signatures in order to determine the success condition for the attacker.

This, however, might not be sufficient. Consider an attacker that queries twice the same message m (through, say, some third party honestly implementing the user's algorithm) and is then able to compute a valid signature on some message $m' \neq m$. Since this adversary queried twice the same message, it *still* has to output three distinct messages in order to succeed in the unforgeability game.

In this section we show that giving the attacker, in addition to controlling the user, access to a protocol oracle (that takes as input a message and returns the signature and the user's transcript) yields a strictly stronger definition.

4.1 Defining honest-user unforgeability

Before proposing the new definition, we fix some notation. Let $\mathcal{P}(sk, pk, \cdot)$ be an oracle that on input a message m executes the signature issue protocol

$\langle \mathcal{S}(sk), \mathcal{U}(pk, m) \rangle$ obtaining a signature σ. Let trans denote the transcript of the messages exchanges in that interaction. We assume that the transcript consists of all messages exchanged between the parties.[6] This oracle then returns (σ, trans).

Definition 5 (Honest-user unforgeability). *An interactive signature scheme* $\mathsf{BS} = (\mathsf{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf})$ *is* honest-user unforgeable *if* Vf *is deterministic and the following holds: For any efficient algorithm $\mathcal{A}$ the probability that experiment* $\mathsf{HUnforge}_{\mathcal{A}}^{\mathsf{BS}}(\lambda)$ *evaluates to 1 is negligible (as a function of λ) where*

```
Experiment $\mathsf{HUnforge}_{\mathcal{A}}^{\mathsf{BS}}(\lambda)$
    $(sk, pk) \leftarrow \mathsf{KG}(1^\lambda)$
    $((m_1^*, \sigma_1^*), \ldots, (m_{k+1}^*, \sigma_{k+1}^*)) \leftarrow \mathcal{A}^{\langle \mathcal{S}(sk), \cdot \rangle^\infty, \mathcal{P}(sk, pk, \cdot)}(pk)$
    Let $m_1, \ldots, m_n$ be the messages queried to $\mathcal{P}(sk, pk, \cdot)$.
    Return 1 iff
        $m_i^* \neq m_j$ for all $i, j$
        $m_i^* \neq m_j^*$ for $i, j$ with $i \neq j$, and
        $\mathsf{Vf}(pk, m_i^*, \sigma_i^*) = 1$ for all $i$, and
        $\mathcal{S}$ has returned $\mathsf{ok}$ in at most $k$ interactions.
```

(When counting the interactions in which $\mathcal{S}$ returns ok*, we do not count the interactions simulated by $\mathcal{P}$.)*

An interactive signature scheme is *strongly honest-user unforgeable* if the condition "$m_i^* \neq m_j$ for all i, j" in the above definition is substituted by "$(m_i^*, \sigma_i^*) \neq (m_j, \sigma_j)$ for all i, j" and if we change the condition "$m_i^* \neq m_j^*$ for i, j with $i \neq j$" to "$(m_i^*, \sigma_i^*) \neq (m_j^*, \sigma_j^*)$ for i, j with $i \neq j$".

Notice that we require Vf to be deterministic. When we drop this requirement, the definition does not behave as one would intuitively expect. We explain this problem in detail in Section 5. Note further that this definition can be further strengthened by giving the adversary also the randomness of the honest user. Notice that all our results and proofs also hold for this stronger definition.

4.2 Unforgeability does not imply honest-user unforgeability

We show that unforgeability does not imply honest-user unforgeability. The high-level idea of our counterexample is to change the verification algorithm of an interactive signature scheme such that it accepts a message m' if it obtains as input two distinct and valid signatures on some message $m \neq m'$ (in addition to accepting honestly generated signatures). More precisely, fix an unforgeable and blind signature scheme $\mathsf{BS} = (\mathsf{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf})$ that is strongly unforgeable. Fix some efficiently computable injective function $f \neq id$ on bitstrings (e.g., $f(m) := 0\|m$). We construct a blind signature scheme $\mathsf{BS}_1 = (\mathsf{KG}_1, \langle \mathcal{S}_1, \mathcal{U}_1 \rangle, \mathsf{Vf}_1)$ as follows:

[6] The definition of honest-user unforgeability could be easily strengthened by including the randomness of $\mathcal{U}$ in trans. Our results also hold with respect to that strengthened definition. However, it is not clear that giving the honest-user's randomness to the adversary models any realistic attacks.

- $\mathsf{KG}_1 := \mathsf{KG}$, $\mathcal{S}_1 := \mathcal{S}$, and $\mathcal{U}_1 := \mathcal{U}$.
- $\mathsf{Vf}_1(pk, m, \sigma)$ executes the following steps:
  - Invoke $v := \mathsf{Vf}(pk, m, \sigma)$. If $v = 1$, return 1.
  - Otherwise, parse σ as (σ^1, σ^2). If parsing fails or $\sigma^1 = \sigma^2$, return 0.
  - Invoke $v_i := \mathsf{Vf}(pk, f(m), \sigma^i)$ for $i = 1, 2$. If $v_1 = v_2 = 1$, return 1. Otherwise return 0.

Lemma 6. *If* BS *is complete, strongly unforgeable, and blind, then* BS_1 *is complete, unforgeable, and blind.*

We omit both the proof of blindness and completeness of BS_1 since they follow directly from the blindness and completeness of BS. The unforgeability follows directly from the unforgeability of the underlying scheme. The main idea behind unforgeability is the following: The only possibility for the adversary to forge a signature is to obtain two different signatures σ_1, σ_2 on the same message $f(m)$. Then (σ_1, σ_2) is a valid signature on m. However, since the underlying scheme BS is strongly unforgeable, the adversary can only get σ_1, σ_2 by performing two signing queries. Thus, using two queries, the adversary gets two signatures on the message $f(m)$ and one on m. This is not sufficient to break the unforgeability of BS_1 since the adversary would need to get signatures on three different messages for that. The full proof is given in [28].

Before proving the next lemma, we need to define what a randomized (interactive) signature is. Roughly speaking, schemes that have this property output the same signature in two independent executions with same message only with negligible probability.

Definition 7 (Randomized signature scheme). *An interactive signature scheme* $\mathsf{BS} = (\mathsf{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf})$ *is* randomized *if with overwhelming probability in* $\lambda \in \mathbb{N}$ *the following holds: for any* (sk, pk) *in the range of* $\mathsf{KG}(1^\lambda)$*, any message* $m \in \{0,1\}^*$*, we have* $\sigma_1 \neq \sigma_2$ *where* $\sigma_1 \leftarrow \langle \mathcal{S}(sk), \mathcal{U}(pk, m) \rangle$ *and* $\sigma_2 \leftarrow \langle \mathcal{S}(sk), \mathcal{U}(pk, m) \rangle$.

Note that any scheme can easily be modified such that is satisfies this definition by letting the user algorithm pick some random value r, setting $m' \leftarrow m \| r$, and by including r in the signature. It is easy to see that, given any randomized interactive signature scheme, we can construct an adversary that queries the oracle $\mathcal{P}$ twice on some message m with $f(m) \neq m$, receives two signatures, $\sigma_1, \neq \sigma_2$ and outputs the pair $(m, (\sigma_1, \sigma_2))$. This pair is a valid forgery for the message $f(m)$ because our adversary has never queried this message to $\mathcal{P}$ and never invoked $\mathcal{S}$ directly. Thus, we immediate get the following lemma (the full proof can be found in [28]).

Lemma 8. *If* BS *is complete and randomized, then* BS_1 *is not honest-user unforgeable.*

By Lemmas 6 and 8 we immediately get:

Theorem 9. *If complete, blind, and strongly unforgeable interactive signature schemes exist, then there are complete, blind, and unforgeable interactive signature schemes that are not honest-user unforgeable.*

Strong honest-user unforgeability The following lemma shows that strong unforgeability implies strong honest-user unforgeability.

Lemma 10. *Assume that* BS *is complete,*[7] *randomized, and strongly unforgeable. Then* BS *is strongly honest-user unforgeable.*

The full proof is delegated to [28]. This lemma shows that for strongly unforgeable schemes, the traditional (non-honest-user) definition of unforgeability is sufficient. Note, however, that most known blind signature schemes (e.g., [20,15,3,26,19,18]) are not strongly unforgeable. It can also easily be shown that strong unforgeability is strictly stronger than honest-user unforgeability. The separating example appends a bit b to the signature that is ignored by the verification algorithm. Then the signature can easily be changed by flipping the bit. Thus honest-user unforgeability lies strictly between unforgeability and strong unforgeability.

5 Probabilistic verification

In this section we show that, if we allow for a probabilistic verification algorithm, both the definition of honest-user unforgeability, as well as the usual definition of unforgeability will consider schemes to be secure that do not meet the intuitive notion of unforgeability.

One may argue that discussing problems in the definition of blind signature schemes in the case of probabilistic verification is not necessary because one can always just use schemes with deterministic verification. We disagree with this point of view: Without understanding why the definition is problematic in the case of probabilistic verification, there is no reason to restrict oneself to schemes with deterministic verification. Only the awareness of the problem allows us to circumvent it. We additionally give a definition that works in the case of probabilistic verification. This is less important than pointing out the flaws, since in most cases one can indeed use schemes with deterministic verification. But there might be (rare) cases where this is not possible (note that no generic transformation outside the random oracle model is known that makes the verification deterministic).

First, we give some intuition for our counterexample and formalize it afterwards. Assume an interactive signature scheme BS_3 that can distinguishes two kinds of signatures: A full-signature that will pass verification with probability 1, and a half-signature that passes verification with probability $\frac{1}{2}$. An honest interaction between the signer $\mathcal{S}_3$ and the user $\mathcal{U}_3$ will always produce

[7] Completeness is actually necessary to show this lemma: For example, let BS' be a scheme derived from a complete and strongly unforgeable scheme BS in the following way: All machines except for the user are the same in BS and BS'. When the user $\mathcal{U}'$ should sign a message m, he signs $m+1$ instead. Since the user does not occur in the definition of strong unforgeability, the strong unforgeability of BS implies the strong unforgeability of BS'. Yet BS' is not strongly honest-user unforgeable: By performing a signature query for m from the user $\mathcal{U}'$, the adversary can get a valid signature for $m+1$.

a full-signature. A malicious user, however, may interact with the signer to get a half-signature for arbitrary messages. Furthermore, the malicious user may, by sending λ half-signatures to the signer (λ is the security parameter) and executing a special command, get two half-signatures instead of one. ("Buy $\lambda + 1$ signatures, get one free.") At the first glance, one would expect that such a scheme cannot be honest-user unforgeable or even unforgeable. But in fact, the adversary has essentially two options: First, he does not request λ half-signatures. Then he will not get a signature for free and thus will not win in the honest-user unforgeability game. Second, he does request λ half-signatures and then performs the extra query and thus gets $\lambda + 2$ half-signatures using $\lambda + 1$ queries. Then, to win, he needs that all $\lambda + 2$ signatures pass verification (since the definition of unforgeability/honest-user unforgeability requires that $\mathsf{Vf}_3(pk, m_i^*, \sigma_i^*)$ evaluates to 1 for all signatures (m_i^*, σ_i^*) output by the adversary) However, since each half-signature passes verification with probability $\frac{1}{2}$, the probability that all signatures pass verification is negligible ($\leq 2^{-\lambda}$). Thus, the adversary does not win, and the scheme is honest-user unforgeable. Clearly, this is not what one would expect; so Definition 5 should not be applied to the case where the verification is probabilistic (and similarly the normal definition of unforgeability should not be applied either in that case).

More precisely, let $\mathsf{BS} = (\mathsf{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf})$ be a randomized, complete, blind, and honest-user unforgeable interactive signature scheme. Let Q be an efficiently decidable set such that the computation of arbitrarily many bitstrings $m \in Q$ and $m' \notin Q$ is efficiently feasible.

We define the scheme $\mathsf{BS}_3 = (\mathsf{KG}_3, \langle \mathcal{S}_3, \mathcal{U}_3 \rangle, \mathsf{Vf}_3)$ as follows:

- $\mathsf{KG}_3 := \mathsf{KG}$.
- $\mathcal{S}_3(sk)$ behaves like $\mathcal{S}(sk)$, except when the first message from the user is of the form $(\texttt{extrasig}, m_1^\circ, \dots, m_\lambda^\circ, \sigma_1^\circ, \dots, \sigma_\lambda^\circ, m_1', \dots, m_q')$ where λ is the security parameter. Then $\mathcal{S}_3$ executes the following steps:
  - Check whether $m_1^\circ, \dots, m_\lambda^\circ \in Q$ are pairwise distinct messages, and for all $i = 1, \dots, q$ we have $m_i' \notin Q$, and for all $i = 1, \dots, \lambda$ we have $\mathsf{Vf}(pk, 1\|m_i^\circ, \sigma_i^\circ) = 1$.[8] If not, ignore the message.
  - If the check passes, run $\langle \mathcal{S}(sk), \mathcal{U}(pk, 1\|m_i') \rangle$ for each $i = 1, \dots, q$, resulting in signatures $\tilde{\sigma}_i$, and set $\sigma_i' := 1\|\tilde{\sigma}_i$.
  - Then $\mathcal{S}_3$ sends $(\sigma_1', \dots, \sigma_n')$ to the user, outputs ok and does not react to any further messages in this session.
- $\mathcal{U}_3(pk, m)$ runs $\sigma \leftarrow \mathcal{U}(pk, 0\|m)$ and returns $0\|\sigma$.
- $\mathsf{Vf}_3(pk, m, \sigma)$ performs the following steps:
  - If $\sigma = 0\|\sigma'$ and $\mathsf{Vf}(pk, 0\|m, \sigma') = 1$, Vf_3 returns 1.
  - If $\sigma = 1\|\sigma'$ and $\mathsf{Vf}(pk, 1\|m, \sigma) = 1$, Vf_3 returns 1 with probability $p := \frac{1}{2}$ and 0 with probability $1 - p$.
  - Otherwise, Vf_3 returns 0.

Lemma 11. *If* BS *is blind and complete, so is* BS_3*.*

[8] Without loss of generality, we assume that the public key *pk* can efficiently be computed from the secret key *sk*.

Proof. Blindness and completeness of BS_3 follow directly from that of BS. The only difference between the schemes is that instead of a message m, a message $0\|m$ is signed and 0 is prepended to the signatures (as long as the user is honest as is the case in the definitions of blindness and completeness).

Lemma 12. *If* BS *is honest-user unforgeable, so is* BS_3.

The proof idea was already explained at the beginning of this section. The complete proof is given in [28].

The following lemma shows that, although BS_3 is honest-user unforgeable (and thus also unforgeable), it should not be considered secure! Namely, an adversary can, given λ queries, produce $\lambda + 1$ message/signature pairs, each of which passes verification with probability $\frac{1}{2}$. In particular in a setting where the machine which verifies the signatures is stateless and where the adversary may thus just resubmit a rejected signature, such signatures are as good as signatures that pass verification with probability 1. Thus, the adversary has essentially forged one signature.

An adversary that queries the signer λ times on distinct messages (from Q) is able to execute the special command that allows to produce an arbitrary number of half-signatures. Thus, we immediate get (see [28] for the full proof):

Lemma 13. *We call* (m, σ) *a half-signature (with respect to some implicit public-key pk) if the probability that* $\mathsf{Vf}(pk, m, \sigma) = 1$ *is* $1/2$. *If* BS *is complete, then for any polynomial p, there is an adversary* $\mathcal{A}$ *that performs* $\lambda + 1$ *interactions with* $\mathcal{S}_3$ *and does not query* $\mathcal{P}$ *and that, with overwhelming probability, outputs* $p(\lambda)$ *half-signatures* $(m_1^*, \sigma_1^*), \dots, (m_{p(\lambda)}^*, \sigma_{p(\lambda)}^*)$ *such that all* m_i^* *are distinct.*

5.1 Adapting the definition

We have shown that, if we allow for a probabilistic verification algorithm in the definition of honest-user unforgeability (and similarly in the definition of unforgeability), schemes that are intuitively insecure will be considered secure by the definition. There are two possible ways to cope with this problem.

The simplest solution is to require that the verification algorithm is deterministic. This is what we did in Section 4.1 (Definition 5). This choice is justified since almost all known blind signature schemes have a deterministic verification algorithm anyway. Thus restricting the verification algorithm to be deterministic may be preferable to getting a more complicated definition.[9]

In some cases, however, it might not be possible to make the verification deterministic. In such cases, it is necessary to strengthen the definition of honest-user unforgeability. Looking back at our counterexample, the problem was the following: If the adversary produces many signatures that each pass verification

[9] Notice that one could weaken the requirement and only require that two invocations of the verification algorithm output the same value with overwhelming probability. This would allow for verification algorithms that essentially compute a deterministic function but have to solve problems in BPP during that computation.

with non-negligible but not overwhelming probability, this is not considered an attack: The probability that all signatures pass verification simultaneously is negligible. In order to fix this problem, we thus need to change the definition in such a way that a signature that is accepted with non-negligible probability is always considered a successful forgery. More precisely, if a signature passes verification at least once when running the verification algorithm a polynomial number of times, then the signature is considered valid. This idea leads to the following definition:

Definition 14 (Honest-user unforgeability with probabilistic verification). *Given a probabilistic algorithm* Vf *and an integer* t*, we define* Vf^t *as follows:* $\mathsf{Vf}^t(pk, m, \sigma)$ *runs* $\mathsf{Vf}(pk, m, \sigma)$ t*-times. If one of the invocations of* Vf *returns* 1*,* Vf^t *returns* 1*. If all invocations of* Vf *return* 0*,* Vf^t *returns* 0*.*

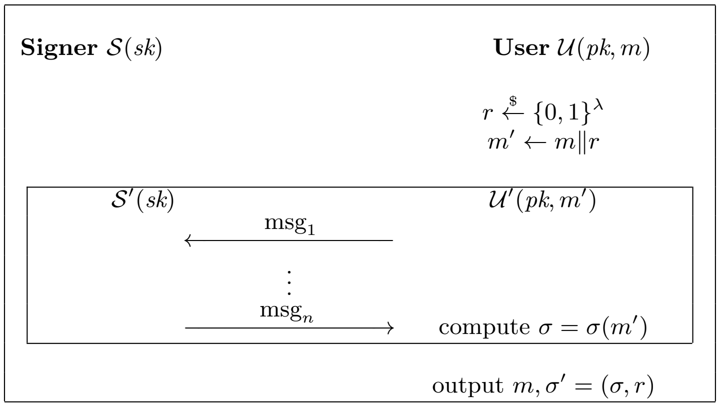
A blind signature scheme $\mathsf{BS} = (\mathsf{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf})$ *is called* honest-user unforgeable (with probabilistic verification) *if the following holds: For any efficient algorithm* $\mathcal{A}$ *and any polynomial* p*, the probability that experiment* $\mathsf{HUnforge}_{\mathcal{A}}^{\mathsf{BS}}(\lambda)$ *evaluates to* 1 *is negligible (as a function of* λ*) where*

```
Experiment $\mathsf{HUnforge}_{\mathcal{A}}^{\mathsf{BS}}(\lambda)$
    $(sk, pk) \leftarrow \mathsf{KG}(1^\lambda)$
    $((m_1^*, \sigma_1^*), \ldots, (m_{k+1}^*, \sigma_{k+1}^*)) \leftarrow \mathcal{A}^{\langle \mathcal{S}(sk), \cdot \rangle^\infty, \mathcal{P}(sk, pk, \cdot)}(pk)$
    Let $m_1, \ldots, m_n$ be the messages queried to $\mathcal{P}(sk, pk, \cdot)$.
    Return 1 iff
        $m_i^* \neq m_j$ for all $i, j$
        $m_i^* \neq m_j^*$ for $i, j$ with $i \neq j$, and
        $\mathsf{Vf}^{p(\lambda)}(pk, m_i^*, \sigma_i^*) = 1$ for all $i$, and
        $\mathcal{S}$ has returned ok in at most $k$ interactions.
```

(When counting the interactions in which $\mathcal{S}$ *returns* ok*, we do not count the interactions simulated by* $\mathcal{P}$*.)*

Notice that the only difference to Definition 5 is that we additionally quantify over a polynomial p, and use $\mathsf{Vf}^{p(\lambda)}$ instead of Vf. If a signature is accepted with non-negligible probability, then there is a polynomial p such that $\mathsf{Vf}^{p(\lambda)}$ will accept that signature with overwhelming probability. (For our counterexample BS_3, one can choose $p(\lambda) := \lambda$ to show that it does not satisfy Definition 14.)

Notice that there is no obvious transformation for taking a signature scheme satisfying the regular unforgeability definition and constructing a scheme secure with respect to Definition 14 out of it. One obvious approach would be to include the randomness for verification in the message and thus to make the scheme deterministic. This might, however, make the scheme totally insecure because in this case a forger might include just the right randomness to get a signature accepted (if that signature would be accepted with negligible but non-zero probability otherwise). Another obvious approach would be to change the verification algorithm such that it verifies each signature p times (for a suitable polynomial p) and only accepts when all verifications succeed. This would make, e.g., half-signatures into signatures with negligible acceptance probability. But



Signer $\mathcal{S}(sk)$
User $\mathcal{U}(pk, m)$
$r \xleftarrow{\$} \{0,1\}^\lambda$
$m' \leftarrow m\|r$
$\mathcal{S}'(sk)$
$\mathcal{U}'(pk, m')$
msg_1
$\vdots$
msg_n
compute $\sigma = \sigma(m')$
output $m, \sigma' = (\sigma, r)$

Fig. 2. Issue protocol of the blind signature scheme

also this approach does not work in general: For any p, the adversary might be able to produce signatures that fails each individual verification with probability $1/2p$ and thus passes the overall verification with constant probability.

6 From unforgeability to honest-user unforgeability

In this section we show how to turn any unforgeable interactive signature scheme into an honest-user unforgeable one. Our transformation is extremely efficient as it only adds some randomness to the message. Therefore, it not only adds a negligible overhead to original scheme, but it also preserves all underlying assumptions. The construction is formally defined in Construction 1 and depicted in Figure 2.

Construction 1 *Let* $\mathsf{BS}' = (\mathsf{KG}', \langle \mathcal{S}', \mathcal{U}' \rangle, \mathsf{Vf}')$ *be an interactive signature scheme and define the signature scheme* BS *through the following three procedures:*

Key Generation. *The algorithm* $\mathsf{KG}(1^\lambda)$ *runs* $(sk', pk') \leftarrow \mathsf{KG}'(1^\lambda)$ *and returns this key-pair.*

Signature Issue Protocol. *The interactive signature issue protocol for message* $m \in \{0,1\}^*$ *is described in Figure 2.*

Signature Verification. *The input of the verification algorithm* Vf *is a public key pk, a message m, and a signature* $\sigma' = (\sigma, r)$*. It sets* $m' \leftarrow (m\|r)$ *and returns the result of* $\mathsf{Vf}'(pk, m\|r, \sigma)$*.*

We first show that our transformation preserves completeness and blindness.

Lemma 15. *If* BS' *is a complete and blind interactive signature scheme, so is* BS*.*

Since the proof follows easily, we omit it here.

Now, we prove that our construction turns any unforgeable scheme into an honest-user unforgeable one.

Lemma 16. *If* BS' *is an unforgeable interactive signature scheme, then* BS *is secure with respect to Definition 5.*

Proof. Assume for the sake of contradiction that BS is not honest-user unforgeable. Then there exists an efficient adversary $\mathcal{A}$ that wins the honest-user unforgeability game with non-negligible probability. We then show how to build an attacker $\mathcal{B}$ that breaks the unforgeability of BS'.

The input of the algorithm $\mathcal{B}$ is a public pk. It runs a black-box simulation of $\mathcal{A}$ and simulates the oracles as follows. Whenever $\mathcal{A}$ engages in an interactive signature issue protocol with the signer, i.e., when the algorithm $\mathcal{A}$ plays the role of the user, then $\mathcal{B}$ relays all messages between $\mathcal{A}$ and the signer. If $\mathcal{A}$ invokes the oracle $\mathcal{P}$ on a message m, then $\mathcal{B}$ picks a random $r \overset{\$}{\leftarrow} \{0,1\}^\lambda$, sets $m' \leftarrow m\|r$, and engages in an interactive signature issue protocol where $\mathcal{B}$ runs the honest user algorithm $\mathcal{U}'$. At the end of this protocol, the algorithm $\mathcal{B}$ obtains a signature σ on the message m'. It sets $\sigma' \leftarrow (\sigma, r)$, stores the pair (m', σ') in a list L and returns σ' together with the corresponding transcript trans to the attacker $\mathcal{A}$.

Eventually, the algorithm $\mathcal{A}$ stops, outputting a sequence of message/signature pairs $(m_1^*, \sigma_1^*), \ldots, (m_{k+1}^*, \sigma_{k+1}^*)$. In this case, $\mathcal{B}$ recovers all message/signature pairs $(m_1', \sigma_1'), \ldots, (m_n', \sigma_n')$ stored in L, it parses σ_i^* as (σ_i', r_i'), it sets $\widetilde{m}_i \leftarrow m_i^*\|r_i^*$ and $\widetilde{\sigma} \leftarrow \sigma_i'$ for all $i = 1, \ldots, k+1$ and outputs $(m_1', \sigma_1'), \ldots, (m_n', \sigma_n')$, $(\widetilde{m}_1, \widetilde{\sigma}_1), \ldots, (\widetilde{m}_{k+1}, \widetilde{\sigma}_{k+1})$.

Analysis. For the analysis first observe that $\mathcal{B}$ runs in polynomial time because $\mathcal{A}$ is efficient and because the handling of all queries can be done efficiently. Suppose that $\mathcal{A}$ succeeds with non-negligible probability. Then it outputs $(k+1)$ message/signature pairs that verify under Vf. Since $\mathcal{B}$ runs the honest user algorithm to compute the signatures $\sigma_1', \ldots, \sigma_n'$ it follows (from the completeness) that all message/signature pairs that $\mathcal{B}$ returns, verify with overwhelming probability. It is left to show that a) the algorithm $\mathcal{B}$ output one more message/signature pair (than queries to the signing oracle with output ok took place) and b) all messages are distinct.

The distinctness property follows immediately from the definition of the success probability in the honest-user unforgeability game and from the construction. More precisely, consider the messages $(m_1', \ldots, m_n')$ and $(\widetilde{m}_1, \ldots, \widetilde{m}_{k+1})$, where $m_i' = m_i\|r_i$ and $\widetilde{m}_j = m_j^*\|r_j^*$. According to our assumption that $\mathcal{A}$ succeeds, it follows that all message pairs m_r^* and m_s^* (for all $r \neq s$) differ from each other. But then it follows easily that $\widetilde{m}_r^*$ and $\widetilde{m}_s^*$ are also distinct (for all $r \neq s$). Since the r_i are chosen randomly, the messages $(m_1', \ldots, m_n')$ also differ from each other with overwhelming probability. Now, consider the messages $(m_1, \ldots, m_n)$ that $\mathcal{A}$ sends to the oracle $\mathcal{P}$. Note that all these messages must differ from the messages $(m_1^*, \ldots, m_{k+1}^*)$ returned by $\mathcal{A}$ by definition. This means, however, that $\widetilde{m}_r^*$ differs from m_i' for all i, r.

Finally we have to show that $\mathcal{B}$ returns one more message/signature pair (property (a)) than protocol executions with the signer $\mathcal{S}'$ took place (and that produced output ok). Since $\mathcal{A}$ wins the game, it follows that in at most k of the protocol executions that $\mathcal{B}$ forwarded between $\mathcal{A}$ and its external signer, the signer returned ok. $\mathcal{B}$ itself has executed n user instances to simulate the oracle $\mathcal{P}$. Since $\mathcal{A}$ outputs $k+1$ message signature pair (s.t. $m_i \neq m_j^*$ for all i, j) it follows that $\mathcal{B}$ has asked at most $n+k$ queries in which the signer $\mathcal{S}'$ returned ok, but $\mathcal{B}$ returned $n+k+1$ message/signature pairs. This, however, contradicts the assumption that BS is unforgeable.

Putting together the above results, we get the following theorem.

Theorem 17. *If complete, blind, and unforgeable interactive signature schemes exist, then there are complete, blind, unforgeable, and honest-user unforgeable interactive signature schemes (with respect to Definition 5).*

The proof of this theorem follows directly from Lemmas 15 and 16.

Acknowledgments. We thank the anonymous reviewers for valuable comments. This research was supported by the Cluster of Excellence "Multimodal Computing and Interaction", by the European Social Fund's Doctoral Studies and Internationalisation Programme DoRa, by the European Regional Development Fund through the Estonian Center of Excellence in Computer Science, EXCS, and by European Social Fund through the Estonian Doctoral School in Information and Communication Technology.

References

1. Michel Abdalla, Chanathip Namprempre, and Gregory Neven. On the (im)possibility of blind message authentication codes. In David Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 262–279, San Jose, CA, USA, February 13–17, 2006. Springer, Berlin, Germany.
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, Berlin, Germany.
3. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 209–236, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Berlin, Germany.
4. Masayuki Abe and Miyako Ohkubo. A framework for universally composable non-committing blind signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 435–450, Tokyo, Japan, December 6–10, 2009. Springer, Berlin, Germany.
5. 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, June 2003.
6. Ronny Bjones. U-prove technology overview. `http://www.itforum.dk/downloads/Ronny_Bjones_Uprove.pdf`, October 2010.
7. 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, Berlin, Germany.

8. Stefan Brands and Christian Paquin. U-prove cryptographic specification v1.0. `http://connect.microsoft.com/site642/Downloads/DownloadDetails.aspx?DownloadID=26953`, March 2011.
9. Stefan A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy.* MIT Press, Cambridge, MA, USA, 2000.
10. Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 08: 15th Conference on Computer and Communications Security*, pages 345–356, Alexandria, Virginia, USA, October 27–31, 2008. ACM Press.
11. 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*, volume 3352 of *Lecture Notes in Computer Science*, pages 134–148, Amalfi, Italy, September 8–10, 2004. Springer, Berlin, Germany.
12. 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.
13. 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.
14. Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 60–77, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Berlin, Germany.
15. Marc Fischlin and Dominique Schröder. Security of blind signatures under aborts. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 297–316, Irvine, CA, USA, March 18–20, 2009. Springer, Berlin, Germany.
16. Marc Fischlin and Dominique Schröder. On the impossibility of three-move blind signature schemes. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 197–215, French Riviera, May 30 – June 3, 2010. Springer, Berlin, Germany.
17. Georg Fuchsbauer. Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Cryptology ePrint Archive, Report 2009/320, 2009. `http://eprint.iacr.org/`.
18. Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh. Round optimal blind signatures. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 630–648. Springer, 2011.
19. E. Ghadafi and N.P. Smart. Efficient two-move blind signatures in the common reference string model. Cryptology ePrint Archive, Report 2010/568, 2010. `http://eprint.iacr.org/`.
20. Carmit Hazay, Jonathan Katz, Chiu-Yuen Koo, and Yehuda Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 323–341, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Berlin, Germany.

21. Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 111–129, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Berlin, Germany.
22. 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, Berlin, Germany.
23. Aggelos Kiayias and Hong-Sheng Zhou. Equivocal blind signatures and adaptive UC-security. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 340–355, San Francisco, CA, USA, March 19–21, 2008. Springer, Berlin, Germany.
24. Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 80–99, New York, NY, USA, March 4–7, 2006. Springer, Berlin, Germany.
25. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
26. Markus Rückert. Lattice-based blind signatures. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 413–430, Singapore, December 5–9, 2010. Springer, Berlin, Germany.
27. Dominique Schröder and Dominique Unruh. Round optimal blind signatures. Cryptology ePrint Archive, Report 2011/264, 2011. `http://eprint.iacr.org/`.
28. Dominique Schröder and Dominique Unruh. Security of blind signatures revisited. Cryptology ePrint Archive, Report 2011/316, 2011. `http://eprint.iacr.org/`.
29. MICROSOFT U-PROVE. Microsoft u-prove ctp release 2. `http://connect.microsoft.com/site642/Downloads/DownloadDetails.aspx?DownloadID=26953`, March 2011.