

Reactively Secure Signature Schemes

Michael Backes, Birgit Pfitzmann, and Michael Waidner

IBM Research, Zurich Research Laboratory
CH-8803 Rüschlikon, Switzerland
{mbc,bpf,wmi}@zurich.ibm.com

Abstract. Protocols for problems like Byzantine agreement, clock synchronization or contract signing often use digital signatures as the only cryptographic operation. Proofs of such protocols are frequently based on an idealizing “black-box” model of signatures. We show that the standard cryptographic security definition for digital signatures is not sufficient to ensure that such proofs are still valid if the idealized signatures are implemented with real, provably secure signatures. We propose a definition of signature security in general reactive, asynchronous environments and prove that for signature schemes where signing just depends on a counter as state the standard security definition implies our definition.

1 Introduction

Protocols for problems like Byzantine agreement, clock synchronization or contract signing often use digital signatures as the only cryptographic operation. Proofs of such protocols typically use a “black-box” model of signatures: they just assume that signatures are unforgeable, i.e., they abstract from all cryptographic details like asymptotic security and error probabilities. Still one should expect that protocols proven secure in this abstract model are also secure if implemented with a real, cryptographically secure signature scheme.

Unfortunately this is not true: Consider an arbitrary, secure signature scheme. We transform such a scheme into a new one which, when signing a message m , attaches to the signature on m all *previously* signed messages and their signatures. This is certainly a strange scheme, but it is easy to see that it satisfies the standard definition of a signature scheme (from [8]), and that it is secure provided the original scheme is secure: A signature scheme is *secure* if there is no polynomial-time (in k , a security parameter) adversary that can produce a forged signature with non-negligible probability. The adversary has exclusive access to a signature oracle, and any signature under a message for which the oracle was not queried counts as a forgery. When asking for the i -th signature the adversary has seen all signatures up to the $(i - 1)$ -st anyway, thus our new scheme is not easier to break than the original one.

Now consider a trivial protocol for fair contract signing: We have two potentially malicious parties A, B and a trusted third party T . Both have inputs c , the contract, and binary values d_X , for $X = A, B$, which tell them whether they should sign ($d_X = 1$) or reject ($d_X = 0$) the contract. The contract should be signed only if all honest parties X start with $d_X = 1$. Now the protocol roughly works like this: Both A and B sign

c , yielding signatures s_A, s_B . If $d_A = 1$ then A stops, and otherwise it sends s_A to T , and similarly B . If T receives both signatures it sends (s_A, s_B) back to A and B , and the contract is considered signed. If T does not receive both signatures (which in an asynchronous network might just mean that T non-deterministically decides to terminate) then T stops and the contract is not signed, which means that nobody should get hold of the pair (s_A, s_B) . Intuitively this protocol is secure, and one can even prove this in the black-box model. But clearly if the protocol is executed multiple times then from each successful run one can construct valid contracts for all previous runs, even for those that did not produce a valid contract.

In Section 2 we introduce the first security definition of digital signature schemes that resolves these problems, and can be used as a basis for “black-box” reasoning about protocols in general asynchronous, reactive environments (see [1] for more details). In Section 3 and 4 we show that for certain signature schemes security in the sense of [8] implies security against this definition. As our example has shown, this cannot be true in general, thus we limit ourselves to schemes where signing needs just a counter as state. This is sufficient for many provably secure signature schemes. For instance, in [8] the signer computes a tree and associates each signature with one of the leaves in this tree. Thus it is sufficient for the signing machine to keep track of which leaves are already used, and this can be easily encoded in a counter. Similar arguments apply, e.g., to the schemes in [10, 5–7].

Similar problems have been already identified for other cryptographic primitives, e.g., oblivious transfer [2] and public-key encryption [3]. For signatures, already the standard definition from [8] is in a reactive setting, but signatures are delivered to the adversary in the same order they were generated. In a general asynchronous setting these orders might be different, which greatly complicates security proofs. In [4], Canetti defines security in a more general reactive setting, but his definition applies to stateless signature algorithms only. This avoids essentially all problems but excludes many provably secure signatures schemes, e.g., [8, 10, 5]. In [9] signatures are implicitly covered within secure channels, but there signatures are use-once, which again avoids all problems but does not lead to a general solution.

2 Definitions and Notation

2.1 Notation

We write “ $:=$ ” for deterministic and “ \leftarrow ” for probabilistic assignment, and “ $\xleftarrow{\mathcal{R}}$ ” for uniform random choice from a set. \downarrow is a distinguished error element available as an addition to the domains and ranges of all functions and algorithms. The fundamental datastructures in our upcoming definitions and proofs are arrays that store tuples of strings. For each array D , the tuples have a predefined structure, e.g., each tuple stores a message along with a signature for it. In order to elegantly capture selection of tuple entries, we adopt some database notation: Entries of a tuple are called *attributes*, e.g., we could have two attributes *msg* and *sig* denoting the message and the signature entry of each tuple. For an element $x \in D$, the value of its attribute *att* is written $x.att$. If the values of one distinguished attribute *att* are unique among all entries in D , i.e., the attribute gives a one-to-one correspondence to the entries of the array, we call *att*

a primary key attribute. We use this to select entries of a tuple, i.e., if a primary key attribute att exists in D and att_2 is another arbitrary attribute in D , we simply write $att_2[a]$ instead of $x.att_2$, where x denotes the unique entry with $x.att = a$. If no such entry exists, we define $att_2[a] := \downarrow$.

2.2 Non-Reactive Definitions

Signature schemes often have memory. As already explained in the introduction, signature schemes that divulge the history of message signed before are not suited for use in an asynchronous reactive environment. In our upcoming definition, we therefore do not allow a signature scheme to use arbitrary parts of its state for signing a message, but we model its memory by a counter.

Definition 1. (*Signature Schemes*) A signature scheme is a triple $(\text{gen}, \text{sign}, \text{test})$ of polynomial-time algorithms, where gen and sign are probabilistic. gen takes an input $(1^k, 1^s)$ with $k, s \in \mathbb{N}$, where s denotes the desired maximum number of signatures, and outputs a pair (sk, pk) of a secret signing key and a public test key in Σ^+ . sign takes such a secret key, a counter $c \in \{1, \dots, s\}$, and a message $m \in \Sigma^+$ as inputs and produces a signature in Σ^+ . We write this $sig \leftarrow \text{sign}_{sk,c}(m)$. Similarly, we write verification as $b := \text{test}_{pk}(m, sig)$ with $b \in \{\text{true}, \text{false}\}$. If the result is true, we say that the signature is valid for m . For a correctly generated key pair, a correctly generated signature for a message m must always be valid for m . \diamond

When we speak about an arbitrary signature scheme in the following, we always mean a counter-based signature scheme in the sense of Definition 1. Signature schemes with truly arbitrary state will not matter in the sequel, and we hence also omit a precise definition.

Security of a signature scheme is defined against existential forgery under adaptive chosen-message attacks:

Definition 2. (*Signature Security*) Given a signature scheme $(\text{gen}, \text{sign}, \text{test})$ and a polynomial $s \in \mathbb{N}[x]$, the signature oracle Sig_s is defined as follows: It has variables sk, pk and a counter c initialized with 0, and the following transition rules:

- First generate a key pair $(sk, pk) \leftarrow \text{gen}(1^k, 1^{s(k)})$, and output pk .
- On input (sign, m) with $m \in \Sigma^+$, and if $c < s(k)$, set $c := c + 1$ and return $sig \leftarrow \text{sign}_{sk,c}(m)$.

The signature scheme is called existentially unforgeable under adaptive chosen-message attack if for every polynomial s and every probabilistic polynomial-time machine A_{sig} that interacts with Sig_s and finally outputs two values m and sig (meant as a forged signature for the message m), the probability that $\text{test}_{pk}(m, sig) = \text{true}$ is negligible (in k) if m is not among the messages previously signed by the signature oracle. \diamond

Lemma 1 (Skipping signatures). Without loss of generality, we can assume that a signature scheme which is secure according to Definition 2 is skipping secure: This

means that the current value of the counter c is not computed within the signature oracle but externally input by the adversary along with the message to be signed. However, Sig_s verifies that the incoming counter values are strictly increasing and do not exceed $s(k)$. \square

Proof. Encode the messages from Σ^+ into Σ^+ such that there is an unused message m^* (e.g., prepend a bit), and let Sig_s sign m^* for a value c if A_{sig} skips it. \blacksquare

2.3 A New Reactive Definition

In order to obtain a security definition that is meaningful in a reactive environment, we have to extend the capabilities of the adversary when interacting with the signature oracle. More precisely, we still have to allow for signing arbitrary messages, but the obtained signatures are stored within the signature oracle and only output upon request of the adversary. Now a signature is considered a forgery for a message m if the signature is valid for m , and if no signature for this particular message has been requested. This is captured in the following definition.

Definition 3. (*Reactive Signature Security*) Given a signature scheme and a polynomial $s \in \mathbb{N}[x]$, the reactive signature oracle RSig_s is defined as follows: It contains variables sk, pk , a counter c initialized with 0, an initially empty set \mathcal{C} of counter values, and an initially empty array SIGS with attributes c, m , and sig for storing counter values, messages, and already made signatures. The counter c can be used as a primary key attribute, which is clear by inspection of the below transitions. The transition rules of RSig_s are:

- First generate a key pair $(sk, pk) \leftarrow \text{gen}(1^k, 1^{s(k)})$, and output pk .
- On input (sign, m) with $m \in \Sigma^+$, and if $c < s(k)$, set $c := c + 1$ and $sig \leftarrow \text{sign}_{sk,c}(m)$, and store (c, m, sig) in SIGS .
- On input (choose, i) , and if $i \leq c$, set $\mathcal{C} := \mathcal{C} \cup \{i\}$ and return $sig[i]$.

The signature scheme is called *reactively secure* if for every polynomial s and every probabilistic polynomial-time machine A_{sig} that interacts with RSig_s and finally outputs two values m and sig (meant as a forged signature for m), the probability that $\text{test}_{pk}(m, sig) = \text{true}$ for $m \neq m[c]$ for all $c \in \mathcal{C}$ is negligible (in k). \diamond

Lemma 2 (Memory-less Schemes). Let Sig denote a memory-less signature scheme, i.e., signing of messages does not depend on prior inputs. If Sig is existentially unforgeable under adaptive chosen message attacks, then it is also reactively secure. \square

Proof. If an adversary breaks Sig in the reactive scenario of Definition 3, we can easily construct an adversary that has the same success probability against the same signature scheme in the non-reactive scenario of Definition 2: This new adversary simply defers signature requests of the original adversary, i.e., inputs of the form (sign, m_j) , until the original adversary chooses those signatures. As the signature oracle is memory-less, making the signatures in the wrong order makes no difference. Moreover, every forged signature by the original adversary is also a suitable forgery for the new adversary. \blacksquare

3 Reduction Proof for Unchanged Signature Schemes

In this section, we show that an arbitrary signature scheme which is secure according to Definition 2 is already reactively secure. This means that signature schemes that only maintain a counter as their local state can safely be used in an asynchronous reactive environment, i.e., without having to bother about problems as sketched in the introduction. As a drawback however, we will see that the concrete complexity gets worse.

Theorem 1. *A signature scheme Sig is reactively secure if and only if it is existentially unforgeable under adaptive chosen-message attack.* \square

Proof. The proof of the left-to-right direction is straightforward, since a reactively secure system is in particular secure for an adversary requesting all signed messages in the correct order, which corresponds to security against existential forgery under adaptive chosen-message attacks.

In order to prove the opposite direction, we show that if there exists a successful adversary Adv^* against the signature scheme in a reactive scenario, there as well exists another adversary Adv for attacking the scheme in a non-reactive scenario. This is shown in Figure 1. The adversary Adv consists of two machines: the adversary Adv^* , which is used “black-box”, and a *simulator* Sim which interacts with Adv^* and also has access to the (non-reactive) signature oracle Sig_s . Intuitively, Sim tries to act like a valid reactive signature oracle for Adv^* . If Adv^* finally outputs a valid forgery for a so-far unsigned message, Sim uses this forgery to successfully attack the non-reactive signature oracle Sig_s .

The proof idea is that Sim does not sign exactly the messages that the adversary Adv^* requests. Instead, it swaps some for random messages. Now there is a chance that the adversary catches the simulator cheating. However, if it does not, we can again use the argument that a successful adversary either forges signatures on messages that were never signed at all, or it is able to guess unknown random values.

Description of the simulator. The simulator Sim maintains a counter c initialized with 0 and an initially empty array $SIGS$ for storing counter values, messages, and signatures obtained from the signature oracle. Again, we use attributes c , m , and sig for these entries, and c is used as a primary key attribute. Sim further maintains an initially empty set \mathcal{C} of counter values corresponding to signatures that have already been requested by the adversary.

In order to allow for a successful cheating, Sim maintains two values c^* , $cnt \in \{0, 1, \dots, s(k)\}$. Below, we let the simulator Sim choose a value $c^* \xleftarrow{\mathcal{R}} \{0, \dots, s(k)\}$ denoting the number of the message where it will cheat. Note that the choice $c^* = 0$ is possible, i.e., Sim does not cheat at all in this case. This is important, since an adversary that requests all signatures would otherwise always catch our simulator cheating, and the proof would fail. Moreover, if the adversary lets a message m be signed twice, but chooses none of these indices and later outputs a signature on m , our simulator would lose. Hence our simulator does not change the c^* -th incoming message, but the c^* -th really different message, and then stick to the change whenever this message re-occurs. The value of cnt is initialized with 0 and represents the number of distinct messages

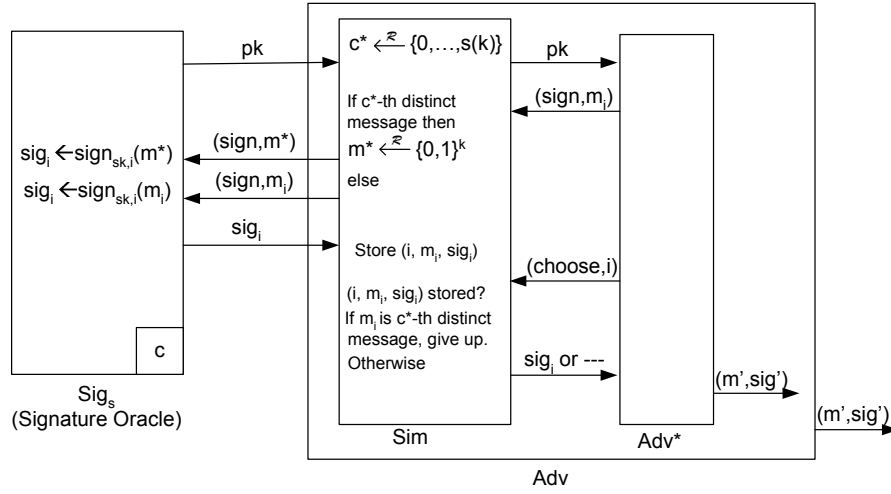


Fig. 1. Overview of the reduction proof for unchanged signature schemes.

that have already been input for signing so far, and it is used to determine when the simulator will cheat for the first time. After this first cheating, the value of cnt will be set to \downarrow and will not matter henceforth.

Furthermore, Sim maintains a variable $m_{ch} \in \Sigma^+$ determining the message that has been discarded for cheating, and a set $c_{ch} \subseteq \{0, \dots, s(k)\}$ containing the counter values c corresponding to the message m_{ch} . (Since the discarded message could be repeatedly input for signing, we have to store several values of c , which illustrate the need for c_{ch} being a set here instead of a single value.) The variables are initialized with \downarrow and $\{\}$ respectively. The behavior of Sim is sketched in Figure 1 and defined as follows:

- First, it chooses $c^* \leftarrow_{\mathcal{R}} \{0, 1, \dots, s(k)\}$.
- On input $(sign, m)$ with $m \in \Sigma^+$: If $c < s(k)$, set $c := c + 1$, otherwise abort, i.e., stop the current transition without any further action. If $m \neq m[i]$ for all $i \leq c$ and $cnt \neq \downarrow$, set $cnt := cnt + 1$. We now distinguish between three cases:
 - (No Cheating): If $cnt \neq c^*$ and $m \neq m_{ch}$, sign in the normal way: Output $(sign, m)$ to the signature oracle yielding a signature sig . Store (c, m, sig) in $SIGS$.
 - (First Cheating): If $cnt = c^*$, let $m^* \leftarrow_{\mathcal{R}} \{0, 1\}^k$ and output $(sign, m^*)$ to the signature oracle yielding a signature sig . Store (c, m^*, sig) in $SIGS$, and set $m_{ch} := m$, $c_{ch} := c_{ch} \cup \{c\}$, and $cnt := \downarrow$.
 - (Repeated Cheating): If $m = m_{ch}$, let $c' \in c_{ch}$ arbitrary, set $m^* := m[c']$ and output $(sign, m^*)$ to the signature oracle yielding a signature sig . Store (c, m^*, sig) in $SIGS$, and set $c_{ch} := c_{ch} \cup \{c\}$.
- On input $(choose, i)$: If $SIGS[i] = \downarrow$ abort. Otherwise let $(i, m, sig) := SIGS[i]$ and set $\mathcal{C} := \mathcal{C} \cup \{i\}$. If $i \in c_{ch}$ then give up the simulation, else output sig to Adv^* .

Proof of Correct Simulation. Assume that Adv^* (reactively) breaks Sig . Thus, in interaction with a correct reactive signature machine RSig_s , it outputs a tuple (m, sig) with $m \neq m[c]$ for all $c \in \mathcal{C}$ and $\text{test}_{pk}(m, \text{sig}^*) = \text{true}$ with non-negligible probability.

The idea is that if the adversary does not choose all of the signed messages, then there is a non-negligible probability that at least one of the remaining messages is the modified one. Provided that the simulator does not explicitly give up the simulation (which happens if the adversary requests a signature for the c^* -th distinct message), we can then show that if Adv^* outputs a valid signature with non-negligible probability, then the probability that this signed message is new, i.e., has not been signed before by the signature oracle, is also non-negligible. In the following, we calculate an upper bound of the probability that our simulation fails to determine a new valid signature. Mainly, there are three possibilities for our simulation to fail:

1. (Simulator gives up): The adversary Adv^* has requested a signature for the cheated message, i.e., an input (choose, i) occurred with $i \in c_{ch}$.
2. (Unsuited signature): The adversary Adv^* outputs a signature for a message m that has been signed before, but not requested by Adv^* .
3. (Guessing the cheated message): The adversary produces a signature for the randomly chosen message m^* .

Moreover, there is a probability of failure, which does not depend on the simulation, but on the fact that the adversary may output either a wrong signature, or a signature for a message that it has already requested. The complemented probability stands for the success probability of the adversary Adv^* if the simulation is done without changing the value of the cheated message. We denote this probability by P_{Adv^*} which is not negligible by assumption.

In the following, let $\varphi(\text{SIGS}) := \{m \mid \exists c, \text{sig}: (c, m, \text{sig}) \in \text{SIGS}\}$ denote the set of all messages that have been signed, and $\varphi(\mathcal{C}) := \{m \mid \exists c \in \mathcal{C}, \text{sig}: (c, m, \text{sig}) \in \text{SIGS}\}$ denote the set of messages that have been chosen by the adversary. Note that the cardinality of these sets denotes the number of distinct messages that have been signed or requested, respectively. As the upcoming calculation of the probability of failure will make extensive use of these cardinalities, we write $\varphi_{\text{sig}} := |\varphi(\text{SIGS})|$ and $\varphi_c := |\varphi(\mathcal{C})|$ for the sake of readability.

For calculating an upper bound for the remaining failure probability, we distinguish between two cases depending on whether Adv^* request all signatures (i.e., $\varphi_c = \varphi_{\text{sig}}$) or not. In the following, we only consider failures because of a giving-up simulator or an unsuited signature. A failure because of guessing the cheated message will be treated separately later on.

- $0 \leq \varphi_c \leq \varphi_{\text{sig}} - 1$: We distinguish two cases:
 - $c_{ch} = \emptyset$: In this case, we obtain a probability of zero that the simulator gives up, and a probability of one for an unsuited signature in the worst case. The probability of $c_{ch} = \emptyset$ is given by $\frac{1}{s(k)+1}$ (representing the case $c^* = 0$) plus $\frac{s(k) - \varphi_{\text{sig}}}{s(k)+1}$ (representing the remaining choices for c^*). In total, the probability of failure for this case is

$$\frac{1 + s(k) - \varphi_{\text{sig}}}{s(k) + 1}.$$

- $c_{ch} \neq \emptyset$: In this case, the simulator gives up with probability $\frac{\varphi_c}{\varphi_{sig}}$, and we obtain an unsuited signature with probability $\frac{\varphi_{sig} - \varphi_c - 1}{\varphi_{sig} - \varphi_c}$. The probability of $c_{ch} \neq \emptyset$ is given by $1 - \frac{1 + s(k) - \varphi_{sig}}{s(k) + 1} = \frac{\varphi_{sig}}{s(k) + 1}$. In total, we obtain a failure probability of

$$\left(\frac{\varphi_c}{\varphi_{sig}} + \frac{\varphi_{sig} - \varphi_c - 1}{\varphi_{sig} - \varphi_c} \right) \frac{\varphi_{sig}}{s(k) + 1} = \frac{\varphi_{sig}^2 - \varphi_{sig} - \varphi_c^2}{(\varphi_{sig} - \varphi_c)(s(k) + 1)}.$$

Joining the probability of both subcases, we obtain a failure probability of

$$\begin{aligned} & \frac{1 + s(k) - \varphi_{sig}}{s(k) + 1} + \frac{\varphi_{sig}^2 - \varphi_{sig} - \varphi_c^2}{(\varphi_{sig} - \varphi_c)(s(k) + 1)} \\ &= \frac{s(k)\varphi_{sig} - \varphi_c - s(k)\varphi_c + \varphi_{sig}\varphi_c - \varphi_c^2}{(\varphi_{sig} - \varphi_c)(s(k) + 1)}. \end{aligned}$$

- $\varphi_c = \varphi_{sig}$: In case $c_{ch} = \emptyset$, we have a probability of zero that the simulator gives up, and also a probability of zero for receiving an unsuited signature since every message has been requested, hence no such message can still be chosen. For the case $c_{ch} \neq \emptyset$, the simulator gives up with probability one, and we obtain an unsuited signature with probability of zero again. The probability of $c_{ch} \neq \emptyset$ is given by $\frac{\varphi_{sig}}{s(k) + 1}$, which is hence also the total probability for this case.

Since the probability in the first case depends on the number φ_c of requested messages, we look for the maximum of all these probabilities when varying φ_c . Differentiation of the formula shows that it reaches its maximum at the border of the interval $[0, \dots, \varphi_{sig} - 1]$ as $\frac{s(k)}{s(k) + 1}$. (The minimum is at $\lceil \varphi_{sig} - \sqrt{\varphi_{sig}} \rceil$). We finally have to maximize the probabilities for the second case when varying φ_{sig} . The formula is strictly increasing, hence it reaches its maximum at $\varphi_{sig} = s(k)$. This yields a probability of $\frac{s(k)}{s(k) + 1}$, which matches the probability of the first case.

So far, we omitted the failures that stem from receiving a forgery for the randomly chosen message m^* . This probability is trivially upper bounded by $\frac{\varphi_{sig}}{2^k}$. Putting it all together, the probability that our simulation is correct is lower bounded by

$$\left(1 - \frac{s(k)}{s(k) + 1} - \frac{\varphi_{sig}}{2^k} \right) \cdot P_{Adv^*} = \left(\frac{1}{s(k) + 1} - \frac{\varphi_{sig}}{2^k} \right) \cdot P_{Adv^*},$$

which is not negligible, since φ_{sig} is upper-bounded by $s(k)$. ■

We need the same number of oracle queries in the proof, but we have a significantly lower probability of success, i.e., the concrete complexity gets worse. A possibility to obtain almost the same probability of success is presented in the next section.

4 Obtaining Reactively Secure Signature Schemes by Additional Randomization

In the following, we show how additional randomization can be used to transform a signature scheme which is secure according to the non-reactive definition 2 into a scheme that is reactively secure with almost the same concrete complexity.

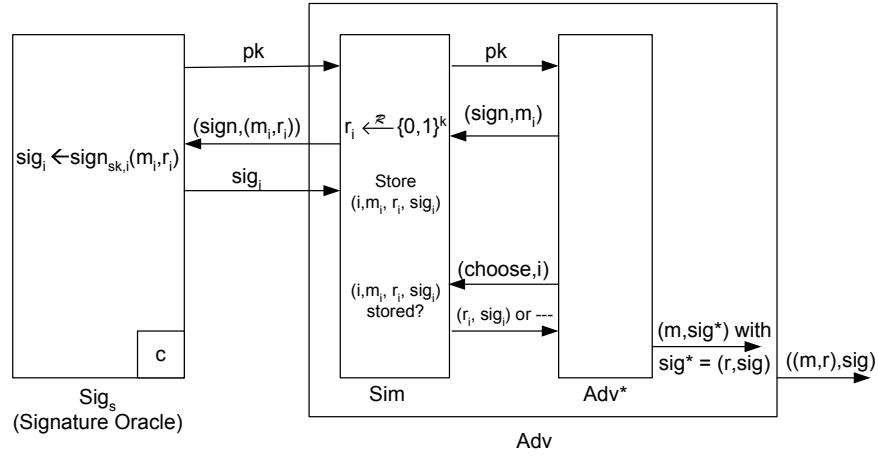


Fig. 2. Overview of the reduction proof for the randomized scheme.

Let $\text{Sig}_1 = (\text{gen}, \text{sign}, \text{test})$ be an arbitrary signature scheme. Then we can transform Sig_1 into another signature scheme $\text{Sig}_2 = (\text{gen}, \text{sign}^*, \text{test}^*)$ as follows: For signing of a message, we define $\text{sign}_{sk,c}^*(m) := (r, \text{sign}_{sk,c}((m, r)))$ for $r \leftarrow_{\mathcal{R}} \{0, 1\}^k$, where we assume that the tuple (m, r) is efficiently encoded into Σ^+ , similar in the following. For testing of a signature and a corresponding message, we define $\text{test}_{pk}^*(m, \text{sig}^*) := \text{test}_{pk}((m, r), \text{sig})$ if sig^* is of the form (r, sig) , and false otherwise. We want to show that if Sig_1 is secure against existential forgery under adaptive chosen message attack, then Sig_2 is reactively secure. This is captured in the following theorem.

Theorem 2. *Let $\text{Sig}_1 = (\text{gen}, \text{sign}, \text{test})$ be a signature scheme, and let $\text{Sig}_2 = (\text{gen}, \text{sign}^*, \text{test}^*)$ be the transformed signature scheme of Sig_1 as defined above. Then Sig_2 is reactively secure if Sig_1 is secure against existential forgery under adaptive chosen message attack. \square*

Proof. We again show that if there exists a successful adversary Adv^* against the signature scheme Sig_2 in a reactive scenario, there as well exists another adversary Adv for attacking the scheme Sig_1 in a non-reactive scenario. The adversary Adv again consists of the adversary Adv^* used “black-box”, and a *simulator* Sim which interacts with Adv^* and also has access to the (non-reactive) signature oracle Sig_s .

Description of the Simulator. The simulator Sim maintains a counter c initialized with 0 and an initially empty set \mathcal{C} of counter values corresponding to signatures that have been requested by the adversary. Finally, Sim maintains an initially empty array SIGS for storing tuples of signatures obtained from the signature oracle Sig_s , along with the corresponding counter, the message, and the random value r . We have attributes sig , c , m , and r for these entries, and c is used as the primary key attribute. The behavior of Sim is sketched in Figure 2 and formally expressed as follows:

- On input (sign, m) with $m \in \Sigma^+$: If $c < s(k)$, set $c := c + 1$ and $r \xleftarrow{\mathcal{R}} \{0, 1\}^k$ and output $(\text{sign}, (m, r))$ to the signature oracle, yielding a signature sig for the tuple (m, r) . Store (c, m, r, sig) in $SIGS$.
- On input (choose, i) : If $SIGS[i] = \perp$ abort. Otherwise let $(i, m, r, sig) := SIGS[i]$, set $\mathcal{C} := \mathcal{C} \cup \{i\}$ and output (r, sig) to Adv^* .

Proof of Correct Simulation. Now assume that Adv^* breaks Sig_2 . Thus, with non-negligible probability, it outputs a tuple (m, sig^*) with $m \neq m[c]$ for all $c \in \mathcal{C}$ and $\text{test}_{pk}^*(m, sig^*) = \text{true}$. The validity of the signature implies that sig^* is of the form $sig^* = (r, sig)$. We distinguish two cases:

1. The pair (m, r) does not occur in $SIGS$ (as second and third elements of a quadruple). In this case, we can use sig as a valid forgery of (m, r) with respect to Sig_1 , since $\text{test}_{pk}^*(m, sig^*) = \text{true}$ implies $\text{test}_{pk}((m, r), sig) = \text{true}$, and the message (m, r) has never been signed by Sig_s .
2. A tuple (c, m, r, sig) occurs in $SIGS$. Then $c \notin \mathcal{C}$ because of the precondition $m \neq m[c]$ for all $c \in \mathcal{C}$. We show that this means that Adv^* has guessed an unknown random value correctly. The inputs $(r[c'], sig[c'])$ that Adv^* obtained do not depend at all on the value of $r[c]$. More precisely, the probability distribution of these inputs is independent of the value $r[c]$, since only the counter of the signature oracle is considered for message signing, and the random value $r[c]$ does not influence this counter. Hence, the probability of guessing $r[c]$ is upper bounded by $\frac{s(k)}{2^k}$. This is negligible since the upper bound $s(k)$ on the number of messages signed is polynomial. ■

In contrast to the results of the previous section, the concrete complexity is almost optimal: We need the same number of oracle queries and obtain almost the same success probability $P_{\text{Adv}} = (1 - \frac{s(k)}{2^k})P_{\text{Adv}^*}$. The computational complexity of Sim is essentially bounded by generating $s(k)$ random values and $s(k)$ lookups in a sorted array of length $s(k)$.

Acknowledgments

We thank an anonymous reviewer for pointing out an improvement to the proof of Theorem 1. We will incorporate this improvement in the long version of this paper.

References

1. M. Backes, B. Pfitzmann, and M. Waidner. A universally composable cryptographic library. IACR Cryptology ePrint Archive 2003/015, Jan. 2003. <http://eprint.iacr.org/>.
2. D. Beaver. How to break a "secure" oblivious transfer protocol. In *Advances in Cryptology: EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 285–296. Springer, 1992.

3. M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology: EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2000.
4. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2001.
5. R. Cramer and I. Damgård. Secure signature schemes based on interactive protocols. In *Advances in Cryptology: CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 297–310. Springer, 1995.
6. R. Cramer and I. Damgård. New generation of secure and practical RSA-based signatures. In *Advances in Cryptology: CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 173–185. Springer, 1996.
7. C. Dwork and M. Naor. An efficient existentially unforgeable signature scheme and its applications. *Journal of Cryptology*, 11(3):187–208, 1998.
8. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
9. B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proc. 22nd IEEE Symposium on Security & Privacy*, pages 184–200, 2001.
10. J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proc. 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 387–394, 1990.