How to Break and Repair a Universally Composable Signature Functionality

Michael Backes[†] and Dennis Hofheinz[⋆]

Abstract. Canetti and Rabin recently proposed a universally composable ideal functionality $\mathcal{F}_{\mathrm{SIG}}$ for digital signatures. We show that this functionality cannot be securely realized by *any* signature scheme, thereby disproving their result that any signature scheme that is existentially unforgeable under adaptive chosen-message attack is a secure realization.
Next, an improved signature functionality is presented. We show that our improved functionality can be securely realized by precisely those signature schemes that are secure against existential forgery under adaptive chosen-message attacks.

Keywords: digital signature schemes, universal composability, adaptive chosen-message attack.

1 Introduction

In this contribution, we investigate the idealization $\mathcal{F}_{\mathrm{SIG}}$ of digital signatures, as defined in [CR02] for the framework of universal composability [Can01]. This framework enjoys a composition theorem which states that specifically, larger protocols may be formulated and investigated by means of an idealization of, e. g., digital signatures, while later a concrete digital signature scheme may be plugged in for the idealization while preserving the security of the large protocol. Certainly, this does not work for any signature scheme, but the scheme must—in a specified sense—*securely realize* the considered idealization, which makes the notion of secure realization (often also called *emulation*) the central notion of the framework.

We show that the idealization $\mathcal{F}_{\mathrm{SIG}}$ cannot be securely realized by *any* real signature scheme. This in particular invalidates the results of [CR02, Claim 2] and [Can01, Claim 14 of full version].[1,2]

† IBM Zurich Research Lab, `mbc@zurich.ibm.com`

⋆ IAKS, Arbeitsgruppe Systemsicherheit, Prof. Dr. Th. Beth, Fakultät für Informatik, Universität Karlsruhe, `hofheinz@ira.uka.de`

1 Our proof applies to the $\mathcal{F}_{\mathrm{SIG}}$-formulation from [CR02] as well as to the slightly older formulation in [Can01].

2 After we had completed and submitted this manuscript, the paper [CR02] was updated; in the updated version [CR03], the functionality $\mathcal{F}_{\mathrm{SIG}}$ was replaced by a functionality $\mathcal{F}_{\mathrm{CERT}}$. Furthermore, a modification of $\mathcal{F}_{\mathrm{SIG}}$—independent of the one described here—was put forward in [Can03]. The generic attack discussed in this paper does not apply to the modified $\mathcal{F}_{\mathrm{SIG}}$ functionality in [Can03].

Next, we propose an improvement of $\mathcal{F}_{\mathrm{SIG}}$ and we show that it can be securely realized by suitable real signature schemes, i.e., by precisely those ones that are secure against existential forgery under adaptive chosen-message attack as defined in [GMR88].

The proof of unrealizability reveals a general problem with detached idealizations of digital signatures: In case of a corrupted signer, signatures for arbitrary messages may be generated *locally* by anyone who has knowledge of the signing key, hence it cannot be guaranteed that the ideal functionality, i. e., the idealization of digital signatures is notified upon every single signature generation. (Consider a larger protocol that honestly generates digital signatures using the publicly distributed signing key of a corrupted signer.) Thus, considering signatures as invalid which are not explicitly "registered" at the ideal functionality causes problems and indeed leads to our attack on $\mathcal{F}_{\mathrm{SIG}}$ discussed below. On the other hand, all signatures not obtained via explicit signing queries to the ideal functionality should intuitively be rejected when they are verified. Our modification of $\mathcal{F}_{\mathrm{SIG}}$ does not have this intuitive rejection property, and in Remark 3, we sketch an approach to milden this problem.

1.1 Overview of this paper

We first briefly review the universal composability framework in Section 2 to prepare the ground for our subsequent results.

In Section 3, we review the ideal signature functionality proposed by Canetti and Rabin and show that it is not securely realizable at all.

In Section 4, we propose an improved functionality for digital signatures, and we show that it can be securely realized precisely by those signature schemes that are existentially unforgeable under adaptive-chosen message attack.

The paper ends with a conclusion (Section 5).

2 Preliminaries

To start, we shortly outline the framework for multi-party protocols as defined in [Can01]. First of all, *parties*, denoted by P_1 through P_n, are modeled as *interactive Turing machines (ITMs)* and are supposed to run some fixed protocol π. There also is an *adversary*, denoted $\mathcal{A}$ and modeled as an ITM as well, which carries out attacks on protocol π. $\mathcal{A}$ may corrupt parties in which case it learns their current and all past states as well as the contents of all their tapes; furthermore, it controls their future actions. $\mathcal{A}$ may further intercept or, when assuming unauthenticated message transfer (which is called the "bare" model in [Can01]), also fake messages sent between parties. If $\mathcal{A}$ corrupts parties only *before* the actual protocol run of π takes place, $\mathcal{A}$ is called *non-adaptive*, otherwise $\mathcal{A}$ is said to be *adaptive*. The respective local inputs for protocol π are supplied by an *environment machine*, which is also modeled as an ITM and denoted $\mathcal{Z}$, that may read all outputs locally made by the parties and communicate with the adversary. Here we only consider environments that guarantee a polynomial (in the security parameter) number of total steps of all participating ITMs. Further discussions on this issue will be given later on, cf. Remark 4.

The model we have just described is called the *real* model of computation. In contrast to this, the *ideal* model of computation is defined just like the real model with the following exceptions. First, we have an additional ITM called the *ideal functionality* $\mathcal{F}$ that is allowed to send messages to and receive messages from the parties privately, i. e., the adversary can neither eavesdrop nor intercept these messages. The ideal functionality cannot be corrupted by the adversary, yet may send messages to and receive messages from it. Secondly, the parties $P_1, \ldots, P_n$ are replaced by *dummy parties* $\tilde{P}_1, \ldots, \tilde{P}_n$ that simply forward their respective inputs to $\mathcal{F}$ and take messages received from $\mathcal{F}$ as output. Finally, the adversary in the ideal model is denoted $\mathcal{S}$ and often called a *simulator*. The attack capabilities of the simulator are restricted to corrupting parties, delaying or even suppressing messages sent from $\mathcal{F}$ to a party, and all actions that are explicitly specified in $\mathcal{F}$. In particular, $\mathcal{S}$ does not have access to the contents of the messages sent from $\mathcal{F}$ to the dummy parties unless the receiving party is corrupted, nor are there any messages actually sent between uncorrupted parties that $\mathcal{S}$ could intercept. Intuitively, the ideal model of computation or, more precisely, the ideal functionality $\mathcal{F}$ itself should represent what we ideally expect a protocol to do. In fact, for a number of standard tasks, there are formulations as such ideal functionalities, e.g., in [Can01].

To decide whether a given protocol π is a secure realization of some ideal functionality $\mathcal{F}$, the framework of [Can01] uses a *simulatability*-based approach: At a time of its choice, $\mathcal{Z}$ may enter its halt state and leave output on its output tape. The probability for $\mathcal{Z}$'s first output bit to be 1 in the real model, when running on security parameter $k \in \mathbb{N}$ and with adversary $\mathcal{A}$ and protocol π is denoted $\mathbf{P}(\mathcal{Z} \to 1 \mid \pi, \mathcal{A})(k)$. The corresponding probability in the ideal model, running with simulator $\mathcal{S}$ and ideal functionality $\mathcal{F}$, is denoted $\mathbf{P}(\mathcal{Z} \to 1 \mid \mathcal{F}, \mathcal{S})(k)$. Now the protocol π is said to *securely realize* the functionality $\mathcal{F}$ if for any adversary $\mathcal{A}$ in the real model, there exists a simulator $\mathcal{S}$ in the ideal model such that for every environment $\mathcal{Z}$, we have that

$$|\mathbf{P}(\mathcal{Z} \to 1 \mid \pi, \mathcal{A})(k) - \mathbf{P}(\mathcal{Z} \to 1 \mid \mathcal{F}, \mathcal{S})(k)| \tag{1}$$

is a negligible[3] function in k.[4] Intuitively, this means that any attack carried out by an adversary in the real model can also be carried out in the idealized modeling with an ideal functionality by a simulator such that no environment is able to tell the difference. In the framework of [Can01], the above definition of security is equivalent to the seemingly weaker requirement that there is a simulator $\mathcal{S}$ so that (1) is a negligible function in k for any environment $\mathcal{Z}$ and the special real-model *dummy adversary* $\tilde{\mathcal{A}}$ which follows explicit instructions from $\mathcal{Z}$.

[3] A function $f\colon \mathbb{N} \to \mathbb{R}$ is called *negligible* if for any $c \in \mathbb{N}$ there exists $k_0 \in \mathbb{N}$ such that $|f(k)| < k^{-c}$ for all $k > k_0$.

[4] The formulation in [Can01] is slightly different but equivalent to the one chosen here which allows to simplify our presentation.

Both the original functionalities $\mathcal{F}_{\mathrm{SIG}}$ of [Can01,CR02] and our variation are *immediate* functionalities. This means that only those simulators are allowed in the ideal model which respect commands from the ideal functionality to deliver a message to a party immediately. This models that only "local" and non-interactive protocols are desired for realizing some functionality. In the case of $\mathcal{F}_{\mathrm{SIG}}$, this intuitively captures that signatures and signature verifications are to be made locally and instantly. Finally and analogously to [Can01], we restrict to *terminating* protocols, i.e., those ones that generate output if all messages are delivered and no party gets corrupted.

Remark 1. In [Can01], the environment machine is modeled as a non-uniform ITM, i. e., as an ITM whose initial input $z = z(k)$ depends on the security parameter k. However, it has been shown in [HMQS03] that the composition theorem of [Can01] remains valid if one restricts to uniform environment machines, i. e., those ones whose initial inputs do not depend on k. Hence it makes sense to alternatively consider only uniform environments where appropriate. In particular, all proofs given below hold for both uniform and non-uniform environments; alone the respective assumptions, i. e., security of a signature scheme with respect to uniform/non-uniform attackers have to be considered with respect to the uniformity class in question.

Remark 2. The modeling of [Can01] does not involve explicit notifications of the ideal functionality upon corruptions of parties. However, a change of model causing ideal functionalities to be informed upon party corruptions was taken in [CK02,CLOS02]. As it is very helpful in our situation and allows for catching adaptively secure realizations of signature schemes, we assume that ideal functionalities are notified upon party corruptions. For our unrealizability result, this makes no difference; see also the remark in the original description of $\mathcal{F}_{\mathrm{SIG}}$ in [CR02].

3 The Attack on the Signature Functionality

In this section, we prove the ideal functionality $\mathcal{F}_{\mathrm{SIG}}$ from [CR02], as depicted in Figure 1, to be unrealizable even with respect to non-adaptive adversaries. In particular, this disproves Claim 2 of [CR02] which asserts any existentially unforgeable signature scheme to be securely realizing $\mathcal{F}_{\mathrm{SIG}}$. Although there is also a formulation of $\mathcal{F}_{\mathrm{SIG}}$ in the full version of [Can01] that slightly differs from the newer formulation in [CR02], we remark that our proof applies without changes also to this former version of $\mathcal{F}_{\mathrm{SIG}}$.

Before we show that $\mathcal{F}_{\mathrm{SIG}}$ cannot be securely realized in general, we point out why the proof of [CR02, Claim 2] goes wrong. The proof is conducted by reduction to the security of the considered signature scheme. It is shown that if an environment $\mathcal{Z}$ existed that distinguished whether it is run with an adversary $\mathcal{A}$ and the signature scheme or with a special simulator $\mathcal{S} = \mathcal{S}(\mathcal{A})$, which is explicitly constructed in [CR02] for any given $\mathcal{A}$, and $\mathcal{F}_{\mathrm{SIG}}$, one could define an attacker G that is able to forge signatures under an adaptive chosen-message

Functionality $\mathcal{F}_{\text{SIG}}$

$\mathcal{F}_{\text{SIG}}$ proceeds as follows, running with parties $P_1, \dots, P_n$ and an adversary $\mathcal{S}$.
Set-up: In the first activation, expect to receive a value (`signer`,sid) from some party P_i. Then, send (`signer`,sid,P_i) to the adversary. From now on, ignore all (`signer`,sid) values.
Signature Generation: Upon receiving a value (`sign`,sid,m) from P_i, hand (`sign`,sid,m) to the adversary. Upon receiving (`signature`,sid,m,σ) from the adversary, set $s_m = \sigma$, send (`signature`,sid,m,σ) to P_i, and request the adversary to deliver this message immediately. Save the pair (m, s_m) in memory.
Signature Verification: Upon receiving a value (`verify`,sid,$P_{i'}$,m,σ) from P_j, do:

1. If $i = i'$ (i. e., the sender identity in the verification request agrees with the identity of the actual signer) then do: If m was never signed then let $f = 0$. If m was signed before (i. e., s_m is defined) and $s_m = \sigma$ then let $f = 1$. If m was signed but $s_m \neq \sigma$ then let the adversary decide on the value of f. (That is, hand (`verify`,sid,P_j,$P_{i'}$,m,σ) to the adversary. Upon receiving $\phi \in \{0, 1\}$ from the adversary, let $f = \phi$.)
2. If $i \neq i'$ then do: If $P_{i'}$ is uncorrupted then set $f = 0$. If $P_{i'}$ is corrupted then let the adversary decide on the value of f, as in Step 1.
3. Once the value of f is set, send (`verified`,sid,m,f) to P_j, and request the adversary to deliver this message immediately.

Fig. 1. The signature functionality $\mathcal{F}_{\text{SIG}}$ reproduced from [CR02]

attack. In the reasoning that G successfully forges signatures, it is argued that "[...] as long as event B does not occur, $\mathcal{Z}$'s view of an interaction with $\mathcal{A}$ and parties running the protocol is distributed identically to its view of an interaction with $\mathcal{S}$[5] and $\mathcal{F}_{\text{SIG}}$ in the ideal process." Here, B denotes the event that during a protocol run in the real model, "[...] $ver(v, m, \sigma) = 1$ for some message m and signature σ, but the signer is uncorrupted and never signed m during the execution of the protocol," where *ver* denotes the signature verification algorithm and v the verification key. This statement is wrong. In case of a corrupted signer, $\mathcal{S}$ takes no actions to signal signatures of messages to $\mathcal{F}_{\text{SIG}}$—however, messages may be signed "legitimately" in case of, e. g., a passively corrupted signer. We will even show below that there is no way to circumvent this problem for the given functionality, since intuitively, there can be no way for $\mathcal{S}$ to determine *which* messages have been signed by an environment taking the role of a corrupted signer. As a consequence, verification requests of such signatures are answered differently in the real model and the ideal one.

In our proof, we use the fact that $\mathcal{F}_{\text{SIG}}$ answers verification requests in the positive only if the corresponding message was already signed *by* $\mathcal{F}_{\text{SIG}}$ *itself.* In Section 4, we will introduce an improved idealization of digital signatures which drops this requirement if the signing party gets corrupted: In case of a corrupted signer, we have to expect even the environment to be able to produce valid signatures for arbitrary messages.

[5] Here we have corrected what seems to be a typo in [CR02].

Theorem 1. *The functionality $\mathcal{F}_{\text{SIG}}$, as specified in [CR02] and depicted in Figure 1, cannot be securely realized by any terminating n-party-protocol π ($n \geq 2$), even when assuming authenticated message transfer and only non-adaptive adversaries.*

Proof. Suppose that a protocol π that has the mentioned termination property securely realizes $\mathcal{F}_{\text{SIG}}$. Fix two different parties P_i and P_j, and also a simulator $\mathcal{S}$ which in the ideal model mimics attacks carried out by the dummy adversary $\tilde{\mathcal{A}}$ on the protocol π. Consider the following environment $\mathcal{Z}_1$, expecting to be run with $\tilde{\mathcal{A}}$ in the real model; of course, in the ideal model, the simulator $\mathcal{S}$ is not bound to these instructions:

1. Activate P_i with input `(signer,`sid`)` and ask the adversary to deliver all messages possibly sent between parties.
2. In the following steps, do *not* request the adversary to deliver any more messages sent between parties.
3. Randomly pick a message r and activate P_i with `(sign,`sid`,`r`)`; extract the signature σ from P_i's answer `(signature,`sid`,`r`,`σ`)`.
4. Activate P_j with input `(verify,`sid`,`P_i`,`r`,`σ`)`; output whatever P_j outputs.

Since π is terminating, we may assume that $\mathcal{Z}_1$ finishes the first of these steps in polynomial time in the real model, and thus in the ideal model. Moreover, the remaining party queries of $\mathcal{Z}_1$ are answered instantly in the ideal model by definition of $\mathcal{F}_{\text{SIG}}$, and thus this also has to hold in the real model. By definition of $\mathcal{F}_{\text{SIG}}$, it follows that $\mathcal{Z}_1$ always outputs 1 if it is run in the ideal model, regardless of the simulator. If it does not do so also in the real model except with negligible probability, $\mathcal{Z}_1$ successfully distinguishes π and $\mathcal{F}_{\text{SIG}}$, which finishes the proof.

Hence assume that $\mathcal{Z}_1$ outputs 1 in the real model with overwhelming probability. Consider the following environment $\mathcal{Z}_2$ that we expect to run with the dummy adversary $\tilde{\mathcal{A}}$ in the real model as well:

1. Tell the adversary to corrupt P_i. Run a local simulation $P_i^{(s)}$ of P_i.
2. Activate $P_i^{(s)}$ with input `(signer,`sid`)` and ask the adversary to deliver—in the name of the corrupted "relay" P_i—all messages sent by the simulated $P_i^{(s)}$ and vice versa.
3. Randomly pick a message r and activate $P_i^{(s)}$ with input `(sign,`sid`,`r`)`, but *suppress* messages sent to or from $P_i^{(s)}$. Wait for $P_i^{(s)}$ to produce output `(signature,`sid`,`r`,`σ`)`.
4. Invoke P_j with input `(verify,`sid`,`P_i`,`r`,`σ`)`; output whatever P_j outputs.

Note that the corruption of P_i is completely passive until step 3. So also $\mathcal{Z}_2$ runs in polynomial time by the termination property of π in the real model, and thus we may assume so also in the ideal model. We analyze the behavior of $\mathcal{Z}_2$ when it is run in the real model. From P_j's point of view, a "regular" run of protocol π exactly as with environment $\mathcal{Z}_1$ takes place. Particularly, since we know that $\mathcal{Z}_1$ outputs 1 with overwhelming probability, we can conclude that σ is accepted by P_j as a valid signature of r in step 4 with overwhelming probability. Consequently, $\mathcal{Z}_2$ will output 1 with overwhelming probability if it is run in the real model.

On the other hand, suppose $\mathcal{Z}_2$ is run in the ideal model with simulator $\mathcal{S}$. Since $\mathcal{S}$ is asked in step 1 to corrupt the signing party P_i, it has the ability to sign message of its choice, i. e., send the corresponding $(\texttt{sign}, sid, m)$ message to $\mathcal{F}_{\mathrm{SIG}}$. However, the randomly chosen message r for which $\mathcal{Z}_2$ locally generates a signature in step 3 cannot be known to $\mathcal{S}$ before step 4. Hence r is registered in $\mathcal{F}_{\mathrm{SIG}}$ as signed (i. e., there is an entry (r, σ) in $\mathcal{F}_{\mathrm{SIG}}$'s list of messages and signatures) in step 4 only with negligible probability. By definition of $\mathcal{F}_{\mathrm{SIG}}$, this means that the uncorrupted party P_j replies in step 4 with 1 only with negligible probability.[6] So in the ideal model, $\mathcal{Z}_2$ outputs 0 with overwhelming probability and therefore distinguishes protocol π and $\mathcal{F}_{\mathrm{SIG}}$. □

4 The Repaired Signature Functionality

We now present a modification of $\mathcal{F}_{\mathrm{SIG}}$, which is realizable even in the bare model by any signature scheme that is existentially unforgeable under adaptive chosen-message attacks. Such schemes exist under reasonable assumptions, e.g., [GMR88,Rom90,CD95,CD96,DN98].

Consider the family of functionalities $\{\mathcal{F}_{\mathrm{SIG}}^{(i)}\}_{P_i}$, where functionality $\mathcal{F}_{\mathrm{SIG}}^{(i)}$ is described in Figure 2. Note that we parameterized the ideal functionality with the identity of the signer. This seems to be necessary since a trivial distinguisher could otherwise activate two parties with input $(\texttt{signer}, sid)$ and then simply check which of these parties is able to actually sign messages. In the ideal model, this is the party that was activated first by definition of $\mathcal{F}_{\mathrm{SIG}}$, yet in the real model this order of activations cannot even be decided by any protocol. We refer to [HMQS03] for a similar problem that arises with the public-key encryption functionality $\mathcal{F}_{\mathrm{PKE}}$. Furthermore, our functionality ensures that signatures never change their validity status which is not always guaranteed in the original formulations of $\mathcal{F}_{\mathrm{SIG}}$. Finally, just like these original formulations, our functionalities $\mathcal{F}_{\mathrm{SIG}}^{(i)}$ are immediate.

Now analogously to the construction in [CR02], we regard a signature scheme $S = (\mathtt{K}, \mathtt{S}, \mathtt{V})$ consisting of probabilistic polynomial-time algorithms for key generation, signing and verifying signatures as a protocol aimed at securely realizing $\mathcal{F}_{\mathrm{SIG}}$. Moreover, if we restrict the execution of $\mathtt{K}$ and $\mathtt{S}$ to a fixed party P_i and to return the verification key upon $(\texttt{signer}, sid)$ requests instead of distributing it, we obtain a protocol called $\pi_S^{(i)}$ that is tailored towards realizing $\mathcal{F}_{\mathrm{SIG}}^{(i)}$. Note

[6] There is a subtlety here: The functionality $\mathcal{F}_{\mathrm{SIG}}$ is not completely specified in [CR02], and it is not clear what happens when signature or verification requests take place without prior "$(\texttt{signer}, \cdot, \cdot)$" initialization, which $\mathcal{S}$ is not forced to send to $\mathcal{F}_{\mathrm{SIG}}$ in the name of an initially corrupted P_i. We find it reasonable to assume that such requests are then ignored; however, our proof applies also to other completions of the specification.

Functionality $\mathcal{F}_{\mathrm{SIG}}^{(i)}$

$\mathcal{F}_{\mathrm{SIG}}^{(i)}$ proceeds as follows, running with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$. (All messages not covered here are simply ignored.)

– Upon receiving `(signer,`sid`)` from P_i (and P_i alone) *for the first time*, send a message `(key,`sid`)` to $\mathcal{S}$; then, upon receiving an answer `(key,`sid`,`v`)` from $\mathcal{S}$, forward this answer to P_i and store v. Ignore further `(signer,`sid`)` requests. *The following rules apply* only *after this initial* `(signer,`sid`)` *message.*
– Upon receiving `(sign,`sid`,`m`)` from P_i (and only P_i), forward this message to $\mathcal{S}$; upon receiving an answer `(signature,`sid`,`m`,`σ`)` from $\mathcal{S}$, forward this answer to P_i. Also store the pair $(m, \sigma, 1)$.
– Upon receiving `(verify,`sid`,`m`,`σ`,`v'`)` from any party P_j, where $v \neq v'$ or v is not determined yet, send this entire tuple to $\mathcal{S}$; upon receiving an answer `(verified,`sid`,`m`,`f`)` from $\mathcal{S}$, forward this answer to P_j.
– Upon receiving `(verify,`sid`,`m`,`σ'`,`v'`)` from any party P_j, where $v = v'$, answer with `(verified,`sid`,`m`,`f`)`, where f is determined as follows:
  - If there is a pair (m, σ, g) with $\sigma = \sigma'$ stored, let $f = g$.
  - If P_i is uncorrupted and there is no pair (m, σ, g) for any σ and g stored, let $f = 0$.
  - In all other cases, send this entire tuple to $\mathcal{S}$; upon receiving an answer `(verified,`sid`,`m`,`f`)` from $\mathcal{S}$, extract f from this answer and store the pair (m, σ', f).

Fig. 2. The modified signature functionality $\mathcal{F}_{\mathrm{SIG}}^{(i)}$

that in contrast to the construction in [CR02], the distribution of the verification key does not have to be covered here, since it is simply a parameter of signature verification requests.

Theorem 2. *For a fixed party P_i, protocol $\pi_S^{(i)}$ securely realizes $\mathcal{F}_{\mathrm{SIG}}^{(i)}$ if and only if the signature scheme S is existentially unforgeable under adaptive chosen-message attacks.*

Proof. For proving the "only if" direction, consider an attacker G that takes part in the following experiment $\mathbf{Exp}_{G,S}^{\text{ef-cma}}(k)$ which is used to define security of digital signature schemes against existential forgery under adaptive chosen-message attack., cf. [GMR88]. Here, $S = (\mathtt{K}, \mathtt{S}, \mathtt{V})$ denotes a signature scheme and $G_k^{\mathtt{S}_s(\cdot)}$ means that G interacts with the corresponding signature oracle with respect to a signing key s.

1. $(s, v) \leftarrow \mathtt{K}(k)$
2. $(m, \sigma) \leftarrow G_k^{\mathtt{S}_s(\cdot)}(v)$
3. **Return** $(\mathtt{V}_v(m, \sigma) \rightarrow \mathtt{accept}) \wedge$ "$\mathtt{S}_s(\cdot)$ was never queried on m in step 2".

Let $\mathcal{Z}$ be the environment that performs the above experiment with a simulated G: $\mathcal{Z}$ triggers G with the public key v gathered through an initial request for key generation to P_i and carries out G's signing requests by redirecting them to P_i in an $\mathcal{F}_{\mathrm{SIG}}^{(i)}$-compatible form. Moreover, the verification request in step 3 is also redirected to P_i. Finally, $\mathcal{Z}$ outputs 1 exactly if the experiment returns `true`.

The simulator $\mathcal{S}_{\mathrm{SIG}}^{(i)}$

– **Communication with $\mathcal{F}_{\mathrm{SIG}}^{(i)}$:**
  - Upon receiving a message $(\texttt{key},sid)$ from $\mathcal{F}_{\mathrm{SIG}}^{(i)}$, run algorithm $\mathtt{K}$ to receive a keypair (s, v); keep s in memory and send $(\texttt{key},sid,v)$ back to $\mathcal{F}_{\mathrm{SIG}}^{(i)}$.
  - Upon receiving $(\texttt{sign},sid,m)$ from $\mathcal{F}_{\mathrm{SIG}}^{(i)}$, let $\sigma \leftarrow \mathtt{S}_s(m)$ and send $(\texttt{signature},sid,m,\sigma)$ back to $\mathcal{F}_{\mathrm{SIG}}^{(i)}$; if s has not yet been initialized (i. e., if there was no former request for key generation), then hand back an error message just as $\pi_S^{(i)}$ would do.
  - Upon receiving $(\texttt{verify},sid,m,\sigma,v')$ from $\mathcal{F}_{\mathrm{SIG}}^{(i)}$, let $f \leftarrow \mathtt{V}_{v'}(m, \sigma)$ and send $(\texttt{verified},sid,m,f)$ back to $\mathcal{F}_{\mathrm{SIG}}^{(i)}$.
  - Deliver all messages sent from $\mathcal{F}_{\mathrm{SIG}}^{(i)}$ to the parties immediately.

– **Communication with $\mathcal{Z}$:**
  - When being requested by $\mathcal{Z}$ to check for messages sent by parties, reply that no messages were sent.
  - When being asked by $\mathcal{Z}$ to deliver a message m to some party P_j, store this request for future use.
  - When being told by $\mathcal{Z}$ to corrupt some party P_j, first corrupt the dummy party $\tilde{P}_j$ to gather information about $\mathcal{Z}$'s communication with $\tilde{P}_j$; then prepare state information for P_j taking into account all of $\tilde{P}_j$'s communication with $\mathcal{Z}$, all messages $\mathcal{S}_{\mathrm{SIG}}^{(i)}$ was asked to deliver to $\tilde{P}_j$ and, if $i = j$, add to this information the signing key s if there was a key generation request and prepare a random tape consistent with the one used by $\mathcal{F}_{\mathrm{SIG}}^{(i)}$ during a possible key generation.

Fig. 3. The simulator $\mathcal{S}_{\mathrm{SIG}}^{(i)}$

Now in the real model, since all requests to P_i are answered "authentically" as it would happen in $\mathbf{Exp}_{G,S}^{\text{ef-cma}}(k)$, the probability that $\mathcal{Z}$ returns 1 is precisely $\mathbf{P}(\mathbf{Exp}_{G,S}^{\text{ef-cma}}(k) \rightarrow \texttt{true})$. On the other hand, $\mathcal{Z}$'s output in the ideal model cannot be 1 at any time regardless of the simulator $\mathcal{S}$ since the condition in step 3 is never fulfilled by definition of $\mathcal{F}_{\mathrm{SIG}}^{(i)}$. So for any simulator $\mathcal{S}$ we have:

$$\left|\mathbf{P}(\mathcal{Z} \rightarrow 1 \mid \pi_S^{(i)}, \tilde{\mathcal{A}})(k) - \mathbf{P}(\mathcal{Z}_k \rightarrow 1 \mid \mathcal{F}_{\mathrm{SIG}}^{(i)}, \mathcal{S})(k)\right| = \mathbf{P}(\mathbf{Exp}_{A,S}^{\text{ef-cma}}(k) \rightarrow \texttt{true}).$$

That means that if G is able to forge signatures under an adaptive chosen-message attack with non-negligible probability, then we can construct an environment $\mathcal{Z}$ that successfully distinguishes $\mathcal{F}_{\mathrm{SIG}}^{(i)}$ and $\pi_S^{(i)}$.

Now for the "if" direction, consider the simulator $\mathcal{S}_{\mathrm{SIG}}^{(i)}$ as described in Figure 3 that mimics attacks carried out by the dummy adversary $\tilde{\mathcal{A}}$ on π. Fix an environment $\mathcal{Z}$. From $\mathcal{Z}$'s point of view, there is no difference between communicating with $\mathcal{S}_{\mathrm{SIG}}^{(i)}$ in the ideal model and talking to $\tilde{\mathcal{A}}$ in the real model by construction of the simulator $\mathcal{S}_{\mathrm{SIG}}^{(i)}$. The only way for $\mathcal{Z}$ to detect a difference between real and ideal model is consequently through requests to parties—note here that the protocol $\pi_S^{(i)}$ does not involve any communication which could possibly be eavesdropped or altered by $\tilde{\mathcal{A}}$.

The following event is defined in analogy to the notation in [CR02]. Let B denote the event that at some point in time at which the signing party P_i is not corrupted, $\mathcal{Z}$ requests any party to verify a valid signature σ of a message m (valid in the sense that $\mathtt{V}(m, \sigma) \to \mathtt{accept}$), where m has not been signed before by an explicit request to P_i. Let $\bar{B}$ denote the event that B does not occur.

We show that by construction of $\mathcal{F}_{\mathrm{SIG}}^{(i)}$ and $\pi_S^{(i)}$, $\mathcal{Z}$'s views in the real and the ideal model do not differ until the event B occurs. If P_i is corrupted, all verification requests of signatures that are not explicitly generated or evaluated by $\mathcal{S}_{\mathrm{SIG}}^{(i)}$ are relayed to it, hence requests are answered exactly as in the real model. On the other hand, if P_i is not corrupted, then the ideal functionality $\mathcal{F}_{\mathrm{SIG}}^{(i)}$ answers on its own only verification requests for signatures that either have been formerly generated or checked by $\mathcal{S}_{\mathrm{SIG}}^{(i)}$, or signatures for messages that have never been signed before. In that case, an answer may only be different to the corresponding answer in the real model if $\mathcal{F}_{\mathrm{SIG}}^{(i)}$ is requested to verify a valid signature of a message that has never been signed before. This is exactly what the event B captures.

It follows that for any fixed security parameter k, we have

$$
\begin{array}{ll}
\textbf{(a)} & \mathbf{P}(B \mid \mathcal{F}_{\mathrm{SIG}}^{(i)}, \mathcal{S}_{\mathrm{SIG}}^{(i)}) \quad = \mathbf{P}(B \mid \pi_S^{(i)}, \tilde{\mathcal{A}}) \\
\textbf{(b)} & \mathbf{P}(\mathcal{Z} \to 1 \mid \bar{B}, \mathcal{F}_{\mathrm{SIG}}^{(i)}, \mathcal{S}_{\mathrm{SIG}}^{(i)}) = \mathbf{P}(\mathcal{Z} \to 1 \mid \bar{B}, \pi_S^{(i)}, \tilde{\mathcal{A}}).
\end{array}
$$

Let G be an attacker on the signature scheme S built from $\mathcal{Z}$ in the following way:

1. Run a simulation of $\mathcal{Z}$.
2. When $\mathcal{Z}$ requests a key generation from party P_i, deliver $\mathcal{Z}$ with the challenge public key v.
3. When $\mathcal{Z}$ asks P_i to sign a message m, redirect this request to the signing oracle $\mathtt{S}_s(\cdot)$.
4. When $\mathcal{Z}$ lets a party P_j verify a signature σ of some message m, compute $f \leftarrow \mathtt{V}_v(m, \sigma)$. If $f = \mathtt{accept}$ and m was not requested to be signed before by the simulated $\mathcal{Z}$, quit the simulation and exit with (m, σ); else answer $\mathcal{Z}$'s request with f.
5. When $\mathcal{Z}$ asks the adversary to report messages sent between parties, reply that no messages were sent; when $\mathcal{Z}$ asks the adversary to deliver a message to a party, ignore this request.
6. When $\mathcal{Z}$ halts or asks the adversary to corrupt P_i, exit with $\mathtt{failed}$.

Taking into consideration **(a)** and **(b)** from above, we obtain for a fixed security parameter k:

$$
\begin{aligned}
& \left| \mathbf{P}(\mathcal{Z} \to 1 \mid \pi_S^{(i)}, \tilde{\mathcal{A}}) - \mathbf{P}(\mathcal{Z} \to 1 \mid \mathcal{F}_{\mathrm{SIG}}^{(i)}, \mathcal{S}_{\mathrm{SIG}}^{(i)}) \right| \\
= & \left| \mathbf{P}(B) \cdot \Big(\mathbf{P}(\mathcal{Z} \to 1 \mid B, \pi_S^{(i)}, \tilde{\mathcal{A}}) - \mathbf{P}(\mathcal{Z} \to 1 \mid B, \mathcal{F}_{\mathrm{SIG}}^{(i)}, \mathcal{S}_{\mathrm{SIG}}^{(i)}) \Big) \right| \\
\leq & \ \mathbf{P}(B) \ = \ \mathbf{P}(\mathbf{Exp}_{G,S}^{\text{ef-cma}} \to \mathtt{true}).
\end{aligned}
$$

Therefore, if $\mathcal{Z}$ successfully distinguishes $\pi_S^{(i)}$ and $\mathcal{F}_{\mathrm{SIG}}^{(i)}$, then G successfully forges signatures. □

Remark 3. We stress that this formulation has a significant drawback compared to the (as we have shown, "over-idealized") functionality $\mathcal{F}_{\mathrm{SIG}}$: In case of a corrupted signer P_i, it is always possible that signature verification requests are answered in the positive, even if $\mathcal{F}_{\mathrm{SIG}}$ itself was not used to sign the message. In view of our proof of Theorem 1, this does not seem to be easily avoidable since in any digital signature scheme, when considered as a non-interactive protocol, the environment can always use the state, i.e., the signing key of a corrupted signer to sign arbitrary messages locally. In a sense, this property of $\mathcal{F}_{\mathrm{SIG}}^{(i)}$ allows for "non-committing" signatures, i.e., with a corrupted signer, it is possible to send signatures whose validity may be determined by the simulator at the time of verification.

One way to milden the "non-committing" property of such signatures is to have the simulator fix a verification algorithm right from the start. More precisely, the simulator $\mathcal{S}$ may be expected to supply the ideal functionality with probabilistic algorithms for signing and signature verification at the beginning of a protocol execution in the ideal model; these algorithms are then executed by the ideal functionality instead of explicitly asking $\mathcal{S}$ for signatures and signature verifications. In this case, the mentioned effect of "non-committing" signatures in case of a corrupted signer is mitigated since the simulator may no longer determine the validity of signatures dynamically, i. e., dependent on data gathered during the protocol run. Moreover, this would circumvent that in the above formalization (just like in the original formulations of $\mathcal{F}_{\mathrm{SIG}}$) the simulator $\mathcal{S}$ is informed about every single message to be signed. Such a modification to $\mathcal{F}_{\mathrm{SIG}}^{(i)}$ could still be shown to yield a security notion equivalent to security under an adaptive chosen-message attack with respect to existential forgeries; in fact, just a small modification to the simulator $\mathcal{S}_{\mathrm{SIG}}^{(i)}$ used in the above proof is necessary to do so. (Note that $\mathcal{S}_{\mathrm{SIG}}^{(i)}$ itself uses only the algorithms $\mathtt{S}$ and $\mathtt{V}$ for signature generation respectively verification.) However, the drawback of such an idealization is that it then itself has to execute cryptographic algorithms, which makes it more difficult to analyze in larger contexts.

Remark 4. In this exposition, certain timing issues have been neglected: For the validity of the composition theorem of [Can01], it is essential that all machines, in particular environment machines and adversaries, underlie computational restrictions. For example it is mandated in [Can01] that all machines only perform a polynomial number of *total* steps. However, to avoid a "trivial" distinction of real and ideal model, it may make more sense to require machines which are polynomially bounded only *per activation*, and to consider only environments explicitly bounded in their total running time, cf. the approach in [HMQS03]. For the environments $\mathcal{Z}_1$ and $\mathcal{Z}_2$ from the proof of our Theorem 1, this is clear by assumption about the protocols π inspected there. Yet, it remains to ensure a polynomial number of steps per activation for our proposed functionalities $\mathcal{F}_{\mathrm{SIG}}^{(i)}$ and the corresponding simulators. One way might be to bound these machines explicitly by a polynomial, thus yielding a *family* of functionalities and simulators. (Note that also the "original" dummy adversary $\tilde{\mathcal{A}}$ from [Can01] is not computationally bounded in any way—so when bounding the simulator, we may assume a bounded real-model adversary, too.) This issue is not treated anywhere in the UC framework, and, given the fundamental importance for this framework, it seems questionable to simply neglect these details in the specification of functionalities and simulators. We refer the reader to the simulatability approach of Pfitzmann et. al. [PW00,PW01,BPW03] for a rigorous treatment of issues of polynomial runtime.

5 Conclusion

We have shown the idealization of signature schemes $\mathcal{F}_{\mathrm{SIG}}$ from [CR02] to be unrealizable, thus invalidating the results of [CR02, Claim 2] and [Can01, Claim 14 of full version]. We have proposed a variant of this digital signature functionality, and we have proven it to be securely realizable by precisely those digital signature schemes that are existentially unforgeable under adaptive chosen-message attacks.

However, we point out that this proof has to be seen in the context of the underlying model: In the present form, the UC framework does not provide a suitable level of detail to allow for rigorous proof techniques to show, e.g., the correctness of a simulator, and issues of polynomial runtime are not treated carefully although they are fundamental for the framework.

References

[BPW03] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A simulatable cryptographic library with nested operations (extended abstract). In *Proc. 10th ACM Conference on Computer and Communications Security (CCS)*, 2003. Extended version in IACR Cryptology ePrint Archive 2003/015, `http://eprint.iacr.org/`.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145. IEEE Computer Society, 2001. Full version at `http://eprint.iacr.org/2000/067`.

[Can03] Ran Canetti. On universally composable notions of security for signature, certification and authentication. Cryptology ePrint archive, November 2003.

[CD95] Ronald Cramer and Ivan 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.

[CD96] Ronald Cramer and Ivan 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.

[CK02] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In *Advances in Cryptology: EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*. Springer, 2002. Full version at `http://eprint.iacr.org/2002/059`.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proc. 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 494–503, 2002. Full version at `http://eprint.iacr.org/2002/140`.

[CR02] Ran Canetti and Tal Rabin. Universal composition with joint state. Cryptology ePrint archive, April 2002. Short version appeared in *Advances in Cryptology: CRYPTO 2003*.

[CR03] Ran Canetti and Tal Rabin. Universal composition with joint state. Cryptology ePrint archive, November 2003.

[DN98] Cynthia Dwork and Moni Naor. An efficient existentially unforgeable signature scheme and its applications. *Journal of Cryptology*, 11(3):187–208, 1998.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[HMQS03] Dennis Hofheinz, Jörn Müller-Quade, and Rainer Steinwandt. On modeling IND-CCA security in cryptographic protocols. Cryptology ePrint Archive, Report 2003/024, February 2003. `http://eprint.iacr.org/2003/024`.

[PW00] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *Proc. 7th ACM Conference on Computer and Communications Security*, pages 245–254, 2000.

[PW01] Birgit Pfitzmann and Michael 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.

[Rom90] John 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.