

Security of Sanitizable Signatures Revisited

Christina Brzuska, Marc Fischlin, Tobias Freudenreich, Anja Lehmann,
Marcus Page, Jakob Schelbert, Dominique Schröder, Florian Volk

Darmstadt University of Technology, Germany
www.minicrypt.de

Abstract. Sanitizable signature schemes, as defined by Ateniese et al. (ESORICS 2005), allow a signer to partly delegate signing rights to another party, called the sanitizer. That is, the sanitizer is able to modify a predetermined part of the original message such that the integrity and authenticity of the unchanged part is still verifiable. Ateniese et al. identify five security requirements for such schemes (unforgeability, immutability, privacy, transparency and accountability) but do not provide formal specifications for these properties. They also present a scheme that is supposed to satisfy these requirements.

Here we revisit the security requirements for sanitizable signatures and, for the first time, present a comprehensive formal treatment. Besides a full characterization of the requirements we also investigate the relationship of the properties, showing for example that unforgeability follows from accountability. We then provide a full security proof for a modification of the original scheme according to our model.

1 Introduction

Sanitizable signature schemes, introduced by Ateniese et al. [1] and, in a slightly different vein, by Steinfeld et al. [2] and Miyazaki et al. [3], allow a signer to delegate signature rights in a controlled way. Namely, the signer can determine parts of the message which a designated party, the sanitizer, can later modify but such that the authenticity and integrity of the remaining parts is still guaranteed. In particular, even the sanitizer should not be able to change inadmissible parts of the message and produce a valid signature for such illegitimate transformations.

A straightforward application of sanitizable signatures are medical data which should be published in an anonymized but authentic form. Suppose for example that for infectious disease surveillance a hospital is obliged to report excerpts of their patients medical data like dates of birth, genders etc. to an authority. Yet other parts of these data can and should be anonymized, e.g., pseudonyms replacing the patients names or deleting psychiatric information.

Ideally, the administrative department of the hospital assembles the requested information from their records, holding the medical data signed by different health professionals, and sanitizes them without further interaction with their personnel. At the same time the authenticity and integrity of the dedicated data should be preserved. Then, clearly, sanitizable signatures in which the hospital

acts as a sanitizer provide a solution. Ateniese et al. [1] provide further applications of sanitizable signature schemes, including multicast, data base outsourcing and secure routing.

Security Requirements. As discussed in [1] meaningful sanitizable signatures come with the usual unforgeability requirement of regular signature schemes:

UNFORGEABILITY. It should be infeasible for an outsider (i.e., neither the signer nor the sanitizer) to forge signatures in the name of the signer or the sanitizer.

But the introduction of the sanitizing party and its relationship to the signer entail further desirable security properties. These are:

IMMUTABILITY. The sanitizer should not be able to produce valid signatures for messages where it has changed other than the designated parts (this can be thought of as an insider attack).

PRIVACY. Sanitized messages and their signatures should not reveal the original data (i.e., the parts which have been sanitized).

TRANSPARENCY. It should be infeasible to decide whether a message has been sanitized or not. This may be desirable in applications where one should not be able to discriminate against messages produced by the sanitizer.

ACCOUNTABILITY. A party (the signer or the sanitizer) should not be held responsible for messages originating from the other party.

While unforgeability can be formalized straightforwardly from the basic case for regular signatures, as it is done in [1], Ateniese et al. remain rather vague when it comes to the other security requirements. Instead, they introduce technical conditions for the sanitizable signature scheme, aiming to achieve the requirements above. Besides unforgeability these are indistinguishability—roughly saying that signatures generated by the signer are computationally independent of the messages—and the property of identical distributions, saying that the signatures produced by the signer and the sanitizer have identical distributions. This approach is arguable in several ways.

First, without having a formal definition of the security requirements above it is hard to tell if a signature scheme with the technical conditions really achieves the desired goals; as always in cryptography, without a robust security model underneath it is impossible to make precise statements about the hardness of attacks. Secondly, having a more abstract view on the desirable security requirements (instead of the scheme’s conditions) facilitates the understanding of their relationships among each other and with other cryptographic primitives. Finally, trying to achieve the security requirements via technical properties seems to be exceedingly restrictive and may exclude otherwise viable solutions.

Our Results. In this paper we revisit the aforementioned security requirements and formalize them according to common game-based approaches. As part of this, we simplify the unforgeability experiment from [1]. We also make several refinements for accountability. First, we augment the model by new algorithms

Proof and **Judge** where **Proof** allows to provide evidence to **Judge** that a message has been sanitized. Then we distinguish between *sanitizer-* and *signer-accountability*, saying that a malicious sanitizer resp. signer cannot falsely accuse the other party. The original approach in [1] only seems to discuss our notion of sanitizer-accountability.

Concerning the relationship of the now-defined security requirements we obtain some useful and also some unexpected results: First, we prove that transparency implies privacy, i.e., any transparent sanitizable signature scheme is also private and for such schemes there is no need to look at the privacy property separately. Secondly, we show that the two accountability types together imply unforgeability, which is in contrast to the position of Ateniese et al. [1] who argue that unforgeability implies accountability. Having a clean model tells us that it is the other way around, and that accountability needs to be considered.

As for the other security properties, immutability, transparency, sanitizer- and signer-accountability we show that each property is independent of the other ones. That is, for each property we present a sanitizable scheme which satisfies all the other requirements except for the one in question. Technically we assume that there are schemes having all properties and then modify the scheme to annihilate the one property. Finally, we show that unforgeability does not follow from sanitizer- or signer-accountability alone (but only if both versions of accountability hold simultaneously). This gives us a complete characterization of the relationship of the notions.

We also revisit the sanitizable signature scheme presented in [1] in light of our formal definitions. We show that a modification of their scheme indeed meets our requirements for immutability, transparency, sanitizer-accountability and signer-accountability. This already implies, via our relationship results, that the scheme is also unforgeable and private and thus a secure sanitizable scheme.

Related Work. As mentioned before, Miyazaki et al. [3] also use the notion of sanitizable signature schemes, but refer to a slightly different approach. According to their notion only deletions of message parts are considered (instead of modifications) and, secondly, the sanitizer is usually not bound to change designated parts of the message but can decide which portions should be deleted. The basic security properties of such sanitizable signature schemes are unforgeability and privacy (following the terminology above). Independently, several similar proposals like content extraction signatures [2] and redactable signatures [4] have been made.

The two approaches for sanitizable signatures and their solutions resemble each other, making the distinction somewhat obscure. This is especially true since further properties have been added to the models in subsequent works, like the requirement that the sanitizer's identity remains hidden [5] in the sanitizable signature model of [3], resembling the above notion of transparency. Nonetheless, one can divide the literature about sanitizable signatures roughly into the works following the approach by Ateniese et al., e.g., [6,7], and the works based on the approach by Miyazaki et al., including [8,9,10,5,11].

We adhere to the notion of sanitizable signature of Ateniese et al. [1], covering message modifications and security requirements like accountability. Some improvements concerning the scheme’s efficiency have been made [6] and some extensions concerning multiple, a-posteriori determined censors have been suggested [7]. None of these proposals goes beyond the original approach to model the security properties formally, though. We note that some of the previous works in the vein of Miyazaki et al. [3] come with security models, especially for privacy and unforgeability [2,5,12]. Yet, they often provide limited security guarantees, like privacy requirements holding for a single message-signature pair only. In contrast our models allow more sophisticated attacks where for instance privacy should still hold for multiple message-signature pairs and even if the attacker can ask for further signatures.

Independently of our work, Yuen et al. [13] also revisit the security of sanitizable signatures, but focus on new constructions.

2 Preliminaries

In this section we define sanitizable signatures. Like a regular signature scheme a sanitizable signature scheme allows to sign messages under the secret signer key sk_{sig} , generated together with the public verification key pk_{sig} . The signing process itself includes a public key pk_{san} of a designated sanitizer and a description ADM of division into blocks and admissible blocks which the sanitizer is allowed to change with the help of its secret key sk_{san} . Any such modification takes the original message and signature and some modification information MOD and produces a signature σ' for the modified message m' .

In the sequel we assume for simplicity that the description ADM of admissible blocks defines the block length $t \in \mathbb{N}$ and contains a set of block numbers from \mathbb{N} which can be changed, and that all messages are aligned to block length (say, by standard padding techniques). The modification information MOD is then a list of pairs $(j, m'[j])$ consisting of a block number j and the new content $m'[j]$ for this block. We say that MOD *matches* ADM if all the block numbers in MOD are admissible according to ADM and the length of the blocks in MOD equals the value in ADM. The case of a more general transformation, where the modifications are modeled as arbitrary algorithms, is straightforward and discussed in Appendix A.

In addition, to settle disputes about the origin of a message-signature pair, an algorithm **Proof** enables the signer to produce a proof π that a signature has been created by the sanitizer. The proof π is generated from a set of previously signed messages. A **Judge** algorithm then uses the proof π to decide if a *valid* message-signature pair (m, σ) has been created by the signer or the sanitizer (the lack of such a proof is interpreted as a signer origin). We note that **Judge** is usually only called for valid pairs (m, σ) ; for invalid pairs settling the dispute is beyond the scheme’s scope.

Definition 1 (Sanitizable Signature Scheme). *A sanitizable signature scheme SanSig consists of seven efficient algorithms $(KGen_{\text{sig}}, KGen_{\text{san}}, \text{Sign}, \text{Sanit}, \text{Verify}, \text{Proof}, \text{Judge})$ such that:*

KEY GENERATION. There are two key generation algorithms, one for the signer and one for the sanitizer. Both create a pair of keys, a private key and the corresponding public key:

$$(pk_{sig}, sk_{sig}) \leftarrow KGen_{sig}(1^n), \quad (pk_{san}, sk_{san}) \leftarrow KGen_{san}(1^n)$$

SIGNING. The *Sign* algorithm takes as input a message $m \in \{0, 1\}^*$, the secret key sk_{sig} of the signer, the public key pk_{san} of the sanitizer as well as a description $ADM \in \mathbb{N} \times 2^{\mathbb{N}}$ of the block length t and admissibly modifiable message blocks from $\{0, 1\}^t$. It outputs a signature (or \perp , indicating an error):

$$\sigma \leftarrow \text{Sign}(m, sk_{sig}, pk_{san}, ADM).$$

We assume that ADM is recoverable from any signature $\sigma \neq \perp$.

SANITIZING. Algorithm *Sanit* takes a message $m \in \{0, 1\}^*$, a signature σ , the public key pk_{sig} of the signer and the secret key sk_{san} of the sanitizer. It modifies the message m according to the modification instruction $MOD \subseteq \mathbb{N} \times \{0, 1\}^t$ (where t is the block length described in ADM) and determines a new signature σ' for the modified message m' . Then *Sanit* outputs m' and σ' (or possibly \perp in case of an error).

$$(m', \sigma') \leftarrow \text{Sanit}(m, MOD, \sigma, pk_{sig}, sk_{san})$$

VERIFICATION. The *Verify* algorithm outputs a bit $d \in \{\text{true}, \text{false}\}$ verifying the correctness of a signature σ for a message m with respect to the public keys pk_{sig} and pk_{san} .

$$d \leftarrow \text{Verify}(m, \sigma, pk_{sig}, pk_{san})$$

PROOF. The *Proof* algorithm takes as input the secret signing key sk_{sig} , a message m and a signature σ as well as a set of (polynomially many) additional message-signature pairs $(m_i, \sigma_i)_{i=1,2,\dots,q}$ and the public key pk_{san} . It outputs a string $\pi \in \{0, 1\}^*$:

$$\pi \leftarrow \text{Proof}(sk_{sig}, m, \sigma, (m_1, \sigma_1), \dots, (m_q, \sigma_q), pk_{san})$$

JUDGE. Algorithm *Judge* takes as input a message m and a valid signature σ , the public keys of the parties and a proof π . It outputs a decision $d \in \{\text{Sig}, \text{San}\}$ indicating whether the message-signature pair has been created by the signer or the sanitizer:

$$d \leftarrow \text{Judge}(m, \sigma, pk_{sig}, pk_{san}, \pi)$$

For a sanitizable signature scheme the usual correctness properties should hold, saying that genuinely signed or sanitized messages are accepted and that a genuinely created proof by the signer leads the judge to decide in favor of the signer.

SIGNING CORRECTNESS. For any security parameter $n \in \mathbb{N}$, any key pair $(sk_{sig}, pk_{sig}) \leftarrow KGen_{sig}(1^n)$, any key pair $(sk_{san}, pk_{san}) \leftarrow KGen_{san}(1^n)$, any message $m \in \{0, 1\}^*$, any $ADM \in \mathbb{N} \times 2^{\mathbb{N}}$ and any $\sigma \leftarrow \text{Sign}(m, sk_{sig}, pk_{san}, ADM)$ we have

$$\text{Verify}(m, \sigma, pk_{sig}, pk_{san}) = \text{true}.$$

SANITIZING CORRECTNESS. For any security parameter $n \in \mathbb{N}$, any key pair $(sk_{\text{sig}}, pk_{\text{sig}}) \leftarrow \text{KGen}_{\text{sig}}(1^n)$, any key pair $(sk_{\text{san}}, pk_{\text{san}}) \leftarrow \text{KGen}_{\text{san}}(1^n)$, any message $m \in \{0, 1\}^*$, any σ with $\text{Verify}(m, \sigma, pk_{\text{sig}}, pk_{\text{san}}) = \text{true}$, any $\text{MOD} \subseteq \mathbb{N} \times \{0, 1\}^t$ matching ADM from σ , and any pair $(m', \sigma') \leftarrow \text{Sanit}(m, \text{MOD}, \sigma, pk_{\text{sig}}, sk_{\text{san}})$ we require

$$\text{Verify}(m', \sigma', pk_{\text{sig}}, pk_{\text{san}}) = \text{true}.$$

PROOF CORRECTNESS. For any security parameter $n \in \mathbb{N}$, any key pair $(sk_{\text{sig}}, pk_{\text{sig}}) \leftarrow \text{KGen}_{\text{sig}}(1^n)$, any key pair $(sk_{\text{san}}, pk_{\text{san}}) \leftarrow \text{KGen}_{\text{san}}(1^n)$, any message $m \in \{0, 1\}^*$, any signature σ , any MOD matching ADM from σ , any $(m', \sigma') \leftarrow \text{Sanit}(m, \text{MOD}, \sigma, pk_{\text{sig}}, sk_{\text{san}})$ with $\text{Verify}(m', \sigma', pk_{\text{sig}}, pk_{\text{san}}) = \text{true}$, and any (polynomially many) m_1, \dots, m_q and $\text{ADM}_1, \dots, \text{ADM}_q$ with $\sigma_i \leftarrow \text{Sign}(m_i, sk_{\text{sig}}, pk_{\text{san}}, \text{ADM}_i)$ and $(m, \sigma) = (m_i, \sigma_i)$ for some i , any $\pi \leftarrow \text{Proof}(sk_{\text{sig}}, m', \sigma', m_1, \sigma_1, \dots, m_q, \sigma_q, pk_{\text{san}})$ we require:

$$\text{Judge}(m', \sigma', pk_{\text{sig}}, pk_{\text{san}}, \pi) = \text{San}.$$

3 Security Requirements

According to Ateniese et al. [1] there are several security requirements that a secure sanitizable signature needs to satisfy. Informally, these are:

UNFORGEABILITY. No outsider should be able to forge the signer's or the censor's signature. This is analogously to the standard security requirement for signatures.

IMMUTABILITY. The censor is allowed to modify predefined, admissible parts of a message, but he should not be able to modify other parts of the message. For example, a sanitizer who is in charge of blackening names in medical documents should not be able to modify the actual medical data.

PRIVACY. Nobody should be able to restore sanitized parts of a message. For example, if we have pseudonyms in medical documents then, of course, the original names should not be recoverable.

TRANSPARENCY. The idea of sanitizable signatures is that, within well-defined limits, the sanitizer inherits the signing authority. Sometimes knowledge of this fact makes the sanitized data less valuable, e.g., an original business plan coming from the CEO is a more desirable target for a spy than the sanitized plan from the administration office. Transparency now says that no one except for the signer and the sanitizer should be able to distinguish signatures from the signer and the sanitizer.

ACCOUNTABILITY. If the signer and the censor have an argument about the origin of a valid message-signature pair (m, σ) , then accountability demands that this dispute can be settled correctly by the Judge. As an example consider a public servant acting as a sanitizer, but publishing unauthorized information in the name of the government.

We next define these notions formally. We note that we call a sanitizable scheme *secure* if it is simultaneously immutable, unforgeable, private, transparent, sanitizer-accountable and signer-accountable according to the definitions below.

We note that our definitions usually consider three parties, the signer, the sanitizer and the adversary (for some properties the adversary takes over the role of one of the other two parties). In practice, though, one usually has many parties, e.g., a sanitizer assigned to many signers. Our definitions are robust in this regard as we leave much power to the adversary and its queries, say, asking the honest signer to sign a message for a chosen public sanitizer key and thus for different sanitizers. By this, our models can be easily mapped to the case of multiple parties by standard guessing strategies (i.e., trying to predict the “target” signer-sanitizer pair and simulating the other honest parties). As an example we show that our notion of immutability also provides security against the “additional sanitizing attack” of [9], a typical non-malleability attack for three parties.

3.1 Unforgeability

The unforgeability notion for sanitizable signatures follows the classical notion for regular signature schemes. It says that nobody should be able to compute a tuple (m^*, σ^*) such that $\text{Verify}(m^*, \sigma^*, pk_{\text{sig}}, pk_{\text{san}}) = \text{true}$ without having the secret keys $sk_{\text{sig}}, sk_{\text{san}}$. This must hold even if one can see additional signatures for other messages. We also give the adversary access to a **Proof** box (as proofs could potentially leak information about the secret signing key). Yet, except for this secret key the adversary fully determines the other input data, including the message-signature pairs and the public keys. This allows to capture for example scenarios where several sanitizers are assigned to the same signer.

Definition 2 (Unforgeability). *A sanitizable signature scheme SanSig is unforgeable if for any efficient algorithm \mathcal{A} the probability that the following experiment returns 1 is negligible (as a function of n):*

Experiment $\text{Unforgeability}_{\mathcal{A}}^{\text{SanSig}}(n)$

$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \text{KGen}_{\text{sig}}(1^n)$
 $(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \text{KGen}_{\text{san}}(1^n)$
 $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{Sign}(\cdot, sk_{\text{sig}}, \cdot), \text{Sanit}(\cdot, \cdot, \cdot, sk_{\text{san}}), \text{Proof}(sk_{\text{sig}}, \dots)}(pk_{\text{sig}}, pk_{\text{san}})$
letting $(m_i, \text{ADM}_i, pk_{\text{san},i})$ and σ_i for $i = 1, 2, \dots, q$
denote the queries and answers to and from oracle **Sign**,
and $(m_j, \text{MOD}_j, \sigma_j, pk_{\text{sig},j})$ and (m'_j, σ'_j) for $j = q + 1, \dots, r$
denote the queries and answers to and from oracle **Sanit**.
return 1 if
 $\text{Verify}(m^*, \sigma^*, pk_{\text{sig}}, pk_{\text{san}}) = \text{true}$ and
for all $i = 1, 2, \dots, q$ *we have* $(pk_{\text{san}}, m^*) \neq (pk_{\text{san},i}, m_i)$ *and*
for all $j = q + 1, \dots, r$ *we have* $(pk_{\text{sig}}, m^*) \neq (pk_{\text{sig},j}, m'_j)$.

3.2 Immutability

The censor can use the Sanit algorithm to change message blocks which the signer declared as modifiable. If a malicious censor tries to modify other blocks this should not yield a correct signature. In the attack model below the malicious sanitizer \mathcal{A} interacts with the signer to receive signatures σ_i for messages m_i , descriptions ADM_i and keys $pk_{\text{san},i}$ of its choice, before eventually outputting a valid pair $(pk_{\text{san}}^*, m^*, \sigma^*)$ such that message m^* is not a “legitimate” transformation of one of the m_i ’s under the same key $pk_{\text{san}}^* = pk_{\text{san},i}$. The latter is formalized by demanding that each m_i and m^* differ in at least one inadmissible block (or that $pk_{\text{san}}^* \neq pk_{\text{san},i}$).

Definition 3 (Immutability). *A sanitizable signature scheme SanSig is immutable if for any efficient algorithm \mathcal{A} the probability that the following experiment returns 1 is negligible (as a function of n):*

Experiment $\text{Immutability}_{\mathcal{A}}^{\text{SanSig}}(n)$

$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \text{KGen}_{\text{sig}}(1^n)$

$(pk_{\text{san}}^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{Sign}(\cdot, sk_{\text{sig}}, \cdot), \text{Proof}(sk_{\text{sig}}, \dots, pk_{\text{sig}}, \cdot)}(pk_{\text{sig}})$

letting $(m_i, \text{ADM}_i, pk_{\text{san},i})$ and σ_i for $i = 1, 2, \dots, q$

denote the queries and answers to and from oracle Sign.

return 1 if

$\text{Verify}(m^*, \sigma^*, pk_{\text{sig}}, pk_{\text{san}}^*) = \text{true}$ and

 for all $i = 1, 2, \dots, q$ we have

$pk_{\text{san}}^* \neq pk_{\text{san},i}$, or

$m^*[j_i] \neq m_i[j_i]$ for some $j_i \notin \text{ADM}_i$

 //where shorter messages are padded with blocks of the special symbol $\perp \notin \{0, 1\}^*$

Thwarting Additional Sanitizing Attacks. Testifying to the fact that our definition is quite robust in the multi-party setting we discuss that our notion of immutability implies the “additional sanitizing attack” of Miyazaki et al. [9]. Suppose we have three parties in a department, the signer and two sanitizers. Both sanitizers are authorized in principle to modify messages, but for a specific message m only the first sanitizer is permitted to do so (say that this message contains information affecting the second sanitizer). Assume now that a requesting party asks for the non-sensitive parts of message m , and that the first sanitizer with public key pk_{san} is honest and changes the message m to derive a new signature σ' for m' . But now the second sanitizer with public key pk_{san}^* intercepts this reply, maliciously deletes the information about him in message m' and produces a signature σ^* for this bowdlerized message m^* . Only this pair m^*, σ^* is sent to the requesting party, looking like an authorized reply to the requesting party.

Our notion of immutability is strong enough to capture “additional sanitizing attacks” (assuming unique public keys of parties). Namely, in our definition we declare the adversary successful if it manages to find a new public key pk_{san}^* different from the sanitizer’s public key pk_{san} such that the final output verifies correctly under this new key pk_{san}^* . An adversary can now mount the additional

sanitizing attack by generating the keys of the honest sanitizer internally (in a sense, giving even more control to the adversary), calling the signer to create the document for the key pk_{san} of the honest sanitizer and then outputting the further censored message m^* with σ^* under a public key pk_{san}^* . Hence, immutability guarantees that such a case cannot succeed and, in particular, that the scheme is secure against “additional sanitizing attacks”.

3.3 Privacy

Privacy roughly means that it should be infeasible to recover information about the sanitized parts of the message. As information leakage through the modified message itself can never be prevented, we only refer to information which is available through the sanitized signature. There are two possible flavors in formalizing privacy for sanitizable signatures. One approach follows semantic security of encryption schemes and is called *semantic privacy*. It says that for any adversary \mathcal{A} seeing sanitized signatures there is a simulator \mathcal{S} which is denied the signatures, but which is still as successful in predicting some information about the original message as \mathcal{A} . This notion is discussed comprehensively in the full version of the paper.

The other approach is based on the indistinguishability notion for encryption. In this case, an adversary can choose pairs (m_0, MOD_0) , (m_1, MOD_1) of messages and modifications together with a description ADM and has access to a “left-or-right” box. This oracle either returns a sanitized signature for the left tuple ($b = 0$) or for the right tuple ($b = 1$). The task of the attacker is to predict the random bit b significantly better than by guessing. Here we need the additional constraint that for each call to the left-or-right box the resulting modified messages are identical for both tuples and the modifications both match ADM, else the task would be trivial. We write $(m_0, \text{MOD}_0, \text{ADM}) \equiv (m_1, \text{MOD}_1, \text{ADM})$ for this.

Below we formalize the more handy indistinguishability notion and discuss in the full paper that the simulation-based approach is equivalent (as in case of encryption). In our definition of privacy we grant the adversary also access to a signature and a sanitizer oracle, enabling the adversary to create signatures which can be sanitized afterwards. We note that the adversary does not get to choose the signature $\sigma_{j,b}$ for inputs to the left-or-right box. Instead, this signature is first computed from scratch. This corresponds to the “hospital setting” mentioned in the introduction, where the medical data and, in particular, their signatures are kept confidentially and only the sanitized document is released. One may define a stronger version where the adversary gets to choose $\sigma_{j,0}, \sigma_{j,1}$, but it seems much harder to realize this requirement efficiently.

As in case of unforgeability and immutability we also grant the adversary access to **Proof**. Hence, since we let the adversary also determine the input to this box the adversary may input the data received from the **Sign** box here, but cannot use any of the initially computed and secret signatures in the calls to the left-or-right box (unless the adversary accidentally guesses one). The reason is again that proofs usually leak information about the signatures but the signatures in the left-or-right box should remain secret (as in the hospital example).

Definition 4 (Privacy). A sanitizable signature scheme SanSig is private if for any efficient algorithm \mathcal{A} the probability that the following experiment returns 1 is negligibly close to $\frac{1}{2}$:

Experiment $\text{Privacy}_{\mathcal{A}}^{\text{SanSig}}(n)$
 $(pk_{sig}, sk_{sig}) \leftarrow KGen_{sig}(1^n)$
 $(pk_{san}, sk_{san}) \leftarrow KGen_{san}(1^n)$
 $b \leftarrow \{0, 1\}$
 $a \leftarrow \mathcal{A}^{\text{Sign}(\cdot, sk_{sig}, \cdot, \cdot), \text{Sanit}(\cdot, \cdot, sk_{san}, \cdot), \text{Proof}(sk_{sig}, \cdot, \cdot), \text{LoRSanit}(\cdot, \cdot, \cdot, sk_{sig}, sk_{san}, b)}(pk_{sig}, pk_{san})$
 where oracle $\text{LoRSanit}(\cdot, \cdot, \cdot, sk_{sig}, sk_{san}, b)$
 on input $(m_{j,0}, \text{MOD}_{j,0}, (m_{j,1}, \text{MOD}_{j,1}))$ and ADM_j
 first computes $\sigma_{j,b} \leftarrow \text{Sign}(m_{j,b}, sk_{sig}, pk_{san}, \text{ADM}_j)$ and then
 returns $(m'_j, \sigma'_j) \leftarrow \text{Sanit}(m_{j,b}, \text{MOD}_{j,b}, \sigma_{j,b}, pk_{sig}, sk_{san})$,
 and where $(m_{j,0}, \text{MOD}_{j,0}, \text{ADM}_j) \equiv (m_{j,1}, \text{MOD}_{j,1}, \text{ADM}_j)$,
 i.e., are mapped to the same modified message.
 return 1 if $a = b$.

3.4 Transparency

For transparency the original work of Ateniese et al. [1] distinguishes between two notions, called weak and strong transparency. In the case of weak transparency an adversary, given a signed message m with a valid signature σ , should not be able to correctly guess whether m has been sanitized or was simply signed. In the case of strong transparency, the adversary should not even be able to tell which parts of the message are potentially mutable. Since the latter seems an overly strong requirement —observe that this implies that the information ADM must be hidden and must not be recoverable from σ , for example— we call weak transparency simply transparency here and formalize only this notion.

We define transparency by the following adversarial game. We consider an adversary \mathcal{A} with access to Sign , Sanit and Proof oracles with which the adversary can create signatures for (sanitized) messages and learn proofs. In addition, \mathcal{A} gets access to a Sanit/Sign box which contains a secret random bit $b \in \{0, 1\}$ and which, on input a message m , a modification information MOD and a description ADM

- for $b = 0$ runs the signer to create $\sigma \leftarrow \text{Sign}(m, sk_{sig}, pk_{sig}, \text{ADM})$, then runs the sanitizer and returns the sanitized message m' with the new signature σ' , and
- for $b = 1$ acts as in the case $b = 0$ but also signs m' from scratch with the signing algorithm to create a signature σ' and returns the pair (m', σ') .

Adversary \mathcal{A} eventually produces an output a , the guess for b . A sanitizable signature is now said to be *transparent* if for all efficient algorithms \mathcal{A} the probability for a right guess $a = b$ in the above game is negligibly close to $\frac{1}{2}$.

Definition 5 (Transparency). A sanitizable signature scheme SanSig is transparent if for any efficient algorithm \mathcal{A} the probability that the following experiment returns 1 is negligibly close to $\frac{1}{2}$:

Experiment $\text{Transparency}_{\mathcal{A}}^{\text{SanSig}}(n)$

$(pk_{sig}, sk_{sig}) \leftarrow \text{KGen}_{sig}(1^n)$
 $(pk_{san}, sk_{san}) \leftarrow \text{KGen}_{san}(1^n)$
 $b \leftarrow \{0, 1\}$
 $a \leftarrow \mathcal{A}^{\text{Sign}(\cdot, sk_{sig}, \cdot), \text{Sanit}(\cdot, \cdot, sk_{san}), \text{Proof}(sk_{sig}, \dots), \text{Sanit/Sign}(\cdot, \cdot, sk_{sig}, sk_{san}, pk_{sig}, pk_{san}, b)}$
with input (pk_{sig}, pk_{san})
where oracle Sanit/Sign for input $m_k, \text{MOD}_k, \text{ADM}_k$
computes $\sigma_k \leftarrow \text{Sign}(m_k, sk_{sig}, pk_{san}, \text{ADM}_k)$
then $(m'_k, \sigma'_k) \leftarrow \text{Sanit}(m_k, \text{MOD}_k, \sigma_k, pk_{sig}, sk_{san})$,
then, if $b = 1$, replaces σ'_k by $\sigma'_k \leftarrow \text{Sign}(m'_k, sk_{sig}, pk_{san}, \text{ADM}_k)$,
and finally returns (m'_k, σ'_k) .
return 1 if $a = b$

We note that, analogously to the case of privacy, we have σ_k be created by the signer locally in the Sanit/Sign box. A stronger requirement would enable the adversary to determine this signature as part of the input. Yet, this notion again does not reflect the “hospital scenario” nor does it seem to be easy to realize efficiently. Similarly, the adversary cannot use these signatures in the Proof box.

Also note that, with the definition above, schemes with deterministic signature or sanitizing algorithms cannot be transparent, because an adversary could then easily compare answers from the Sanit/Sign box with outputs of the signature sanitizing oracle. Yet, since some applications may need transparency even if a message has been signed or sanitized before, we provide the stronger requirement. The weaker guarantee would then also demand from the adversary’s queries to the signing and sanitizing boxes that for all k we have $m'_k \neq m_i$ for all i and $m'_k \neq m'_j$ for all j .

3.5 Accountability

Accountability says that the origin of a (sanitized) signature should be undeniable. There are two types of accountability:

SANITIZER-ACCOUNTABILITY. If a message has not been signed by the signer, then even a malicious sanitizer should not be able to make the judge accuse the signer.

SIGNER ACCOUNTABILITY. If a message and its signature have not been sanitized, then even a malicious signer should not be able to make the judge accuse the sanitizer.

Both notions are formalized below through two similar, yet slightly different adversarial games.

In the sanitizer-accountability game let $\mathcal{A}_{\text{Sanit}}$ be an efficient adversary playing the role of the malicious sanitizer. Adversary $\mathcal{A}_{\text{Sanit}}$ has access to a Sign

oracle and a Proof oracle. Its task is to output a valid message-signature pair m^*, σ^* together with a key pk_{san}^* (with (pk_{san}^*, m^*) being different from messages previously signed by the Sign oracle) such that the proof produced by the signer via Proof still leads Judge to decided “Sig”, i.e., that the signature has been created by the signer.

Definition 6 (Sanitizer-Accountability). A sanitizable signature scheme SanSig is sanitizer-accountable if for any efficient algorithm $\mathcal{A}_{\text{Sanit}}$ the probability that the following experiment returns 1 is negligible (as a function of n):

Experiment San-Accountability $_{\mathcal{A}_{\text{Sanit}}}^{\text{SanSig}}(n)$

$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow KGen_{\text{sig}}(1^n)$
 $(pk_{\text{san}}^*, m^*, \sigma^*) \leftarrow \mathcal{A}_{\text{Sanit}}^{\text{Sign}(\cdot, sk_{\text{sig}}, \cdot, \cdot), \text{Proof}(sk_{\text{sig}}, \dots)}(pk_{\text{sig}})$
 letting $(m_i, \text{ADM}_i, pk_{\text{san}, i})$ and σ_i for $i = 1, 2, \dots, q$
 denote the queries and answers to and from oracle Sign
 $\pi \leftarrow \text{Proof}(sk_{\text{sig}}, m^*, \sigma^*, (m_1, \sigma_1), \dots, (m_q, \sigma_q), pk_{\text{san}})$
 return 1 if
 $(pk_{\text{san}}^*, m^*) \neq (pk_{\text{san}, i}, m_i)$ for all $i = 1, 2, \dots, q$, and
 $\text{Verify}(m^*, \sigma^*, pk_{\text{sig}}, pk_{\text{san}}^*) = \text{true}$, and
 $\text{Judge}(m^*, \sigma^*, pk_{\text{sig}}, pk_{\text{san}}^*, \pi) = \text{Sig}$

In the signer-accountability game a malicious signer $\mathcal{A}_{\text{Sign}}$ gets a public sanitizing key pk_{san} as input. It is allowed to query a sanitizing oracle about tuples $(m_i, \text{MOD}_i, \sigma_i, pk_{\text{sig}, i}')$ receiving answers (m'_i, σ'_i) . Adversary $\mathcal{A}_{\text{Sign}}$ finally outputs a tuple $(pk_{\text{sig}}^*, \pi^*, m^*, \sigma^*)$ and is considered to succeed if Judge accuses the sanitizer for the new key-message pair pk_{sig}^*, m^* with a valid signature σ^* . Note that our model allows the proof π to contain information about the original message.

Definition 7 (Signer-Accountability). A sanitizable signature scheme SanSig is signer-accountable if for any efficient algorithm $\mathcal{A}_{\text{Sign}}$ the probability that the following experiment returns 1 is negligible (as a function of n):

Experiment Sig-Accountability $_{\mathcal{A}_{\text{Sign}}}^{\text{SanSig}}(n)$

$(pk_{\text{san}}, sk_{\text{san}}) \leftarrow KGen_{\text{san}}(1^n)$
 $(pk_{\text{sig}}^*, \pi^*, m^*, \sigma^*) \leftarrow \mathcal{A}_{\text{Sign}}^{\text{Sanit}(\cdot, \cdot, \cdot, sk_{\text{san}})}(pk_{\text{san}})$
 letting (m'_i, σ'_i) for $i = 1, 2, \dots, q$
 denote the answers from oracle Sanit.
 return 1 if
 $(pk_{\text{sig}}^*, m^*) \neq (pk_{\text{sig}, i}, m'_i)$ for all $i = 1, 2, \dots, q$, and
 $\text{Verify}(m^*, \sigma^*, pk_{\text{sig}}^*, pk_{\text{san}}) = \text{true}$ and
 $\text{Judge}(m^*, \sigma^*, pk_{\text{sig}}^*, pk_{\text{san}}, \pi^*) = \text{San}$

4 Relationships of the Security Requirements

In this section we show that except for the privacy and the unforgeability requirement all other notions are independent (in the sense that none of them follows

from the other properties, even if they all hold at the same time). We first show that privacy follows from transparency alone, and unforgeability holds if the two versions of accountability hold simultaneously. We then show the independence of the other requirements.

We stress again that our results are in contrast to the claim by Ateniese et al. [1] that, for example, accountability follows from the unforgeability requirement. Our results show that unforgeability follows from accountability whereas the other direction is not true. It is not clear if Ateniese et al. [1] consider signer-accountability at all, or merely refer to sanitizer-accountability. However, as we have argued both versions of accountability are desirable to avoid framing attacks from either side, and in either case we also show that sanitizer-accountability alone does not imply unforgeability.

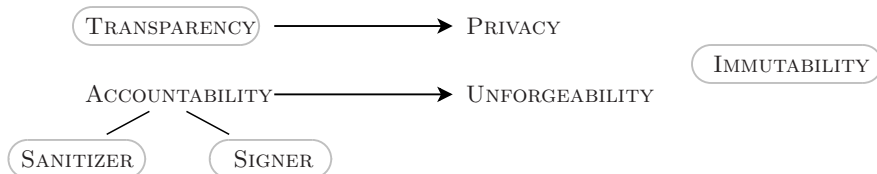


Fig. 1. Summary of the relations among the security properties of sanitizable signatures. Arrows represent implications, frames represent the independence from other requirements.

Implications. We show that privacy follows from transparency. The idea is that for a transparent scheme one cannot distinguish between signatures created by the signer and ones produced by the sanitizer. Hence, we can essentially replace the left-or-right sanitizing oracle in the privacy experiment by the procedure which creates the signatures for the sanitized message with the help of the signer algorithm. But since the privacy experiment requires the sanitized messages to be identical, the answer is always a fresh signature for the same message, independent of the left-or-right question, and privacy follows.

As mentioned above, unforgeability is implied by the two versions of accountability. The idea behind the result is that, given a successful forgery, the judge cannot really decide if this forgery has been produced by the signer or the sanitizer. Else the judge was biased towards outputting `Sig` or `San` for indecisive cases too often, contradicting either the sanitizer- or signer-accountability.

Separations. We further show that all the other security requirements are independent, i.e., no property follows from a combination of the other properties. Our results all assume that there exist secure sanitizable signature scheme obeying all properties (which, according to the next section, exist under common

cryptographic assumptions) and then show that there is a scheme inheriting all properties except for the one in question.

The proofs of the stated implications and separations appear in the full version. We note that this gives a full characterization of the security requirements.

5 Sanitizable Signatures based on Chameleon Hashes

In this section we show that our security requirements can be met. Our construction is a modification of the scheme by Ateniese et al. [1] and also uses chameleon hashes. The idea is as follows: Instead of signing the full message in clear we first replace modifiable message blocks $m[i]$ by (randomized) hash values $h[i] = \text{CHash}(pk_{\text{san}}, m[i]; r[i])$ of the blocks. Then we sign this sequence of message blocks and hash values with a regular signature scheme.

The hash values have the special “chameleon” property that, if one has the sanitizer’s trapdoor information sk_{san} and $r[i]$, one can easily find collisions, i.e., for given $m'[i]$ one is able to determine $r'[i]$ with $h[i] = \text{CHash}(pk_{\text{san}}, m'[i]; r'[i])$, leaving the hash value invariant. This allows the sanitizer to modify message blocks for which the signer includes the $r[i]$ ’s in the signature (and only those), and such that the actual signature on the hash values does not need to be modified. We note that implementing the idea is more complicated due to the accountability problem, requiring something related to (but not exactly like) key-exposure freeness [14] from the chameleon hash. The latter also necessitates the usage of tags entering the hash computations.

5.1 Construction

A chameleon hash scheme $\mathcal{CH} = (\text{CHKGen}, \text{CHash}, \text{CHAdapt})$ (with tags) consists of three efficient algorithms such that algorithm CHKGen on input 1^n returns a key pair (sk, pk) , algorithm CHash on input pk , a tag $\text{TAG} \in \{0, 1\}^n$, a message m and randomness r (which is efficiently samplable from some range \mathcal{R}_{pk}) returns a hash value $h = \text{CHash}(pk, \text{TAG}, m; r)$ and algorithm CHAdapt on input sk, TAG, m, r and TAG', m' returns r' such that $\text{CHash}(pk, \text{TAG}, m; r) = \text{CHash}(pk, \text{TAG}', m'; r')$. It also holds that for any $pk, \text{TAG}, m, \text{TAG}', m'$ the distribution of $\text{CHAdapt}(sk, \text{TAG}, m, r, \text{TAG}', m')$ (over the choice of r) is the same as the distribution of r itself, also implying that a hash value $\text{CHash}(pk, \text{TAG}, m; r)$ (over the choice of r) is distributed independently of TAG, m .

Key-exposure freeness [14] now says that it is infeasible to find collisions, even if one gets to see collisions for other values. To be more precise, the security requirement demands that, after having learned collisions for some tags, one cannot create a collision for a new tag. This is a strong and useful notion and, yet, it would not be sufficient to provide security in our setting. Suppose we attach tags to the documents such that the signer modifies messages by finding collisions for the hash value for the corresponding tags. Then a malicious signer could still try to escape accountability by finding further collisions *for the same tag*.

We therefore introduce the notion of collision-resistance *under random-tagging attacks*, i.e., where collisions for different tags are created but where one of the two tags is chosen at random (and the other one is provided by the adversary). In the full version we show that such chameleon hashes exist under the RSA assumption in the random oracle model:

Definition 8 (Collision-Resistance under Random-Tagging Attacks).

A chameleon hash scheme $\mathcal{CH} = (\text{CHGen}, \text{CHash}, \text{CHAdapt})$ is collision-resistant under random-tagging attacks if for any efficient adversary \mathcal{A} the following experiment returns 1 with negligible probability only:

Experiment $\text{RndTag}_{\mathcal{A}}^{\mathcal{CH}}(n)$

$(pk, sk) \leftarrow \text{CHGen}(1^n)$

$(\text{TAG}, m, r, \text{TAG}', m', r') \leftarrow \mathcal{A}^{\text{OAdapt}(sk, \cdot, \cdot, \cdot)}(pk)$

where oracle OAdapt for the i -th query $(\text{TAG}_i, m_i, r_i, m'_i)$

with $\text{TAG}_i \in \{0, 1\}^n$ samples $\text{TAG}'_i \leftarrow \{0, 1\}^n$ and

computes $r'_i \leftarrow \text{CHAdapt}(sk, \text{TAG}, m, r, \text{TAG}', m')$.

Return (TAG'_i, r'_i) .

return 1 if

$(\text{TAG}, m) \neq (\text{TAG}', m')$ and

$\text{CHash}(pk, \text{TAG}, m; r) = \text{CHash}(pk, \text{TAG}', m'; r')$ and

$\{(\text{TAG}, m), (\text{TAG}', m')\} \neq \{(\text{TAG}_i, m_i), (\text{TAG}'_i, m'_i)\}$ for $i = 1, 2, \dots$ and

$\{(\text{TAG}, m), (\text{TAG}', m')\} \neq \{(\text{TAG}'_i, m'_i), (\text{TAG}'_j, m'_j)\}$ for $i, j = 1, 2, \dots$

The condition $\{(\text{TAG}, m), (\text{TAG}', m')\} \neq \{(\text{TAG}_i, m_i), (\text{TAG}'_i, m'_i)\}$ rules out trivial duplication attacks in which the adversary simply copies the data from the interaction with the oracle. The other condition $\{(\text{TAG}, m), (\text{TAG}', m')\} \neq \{(\text{TAG}'_i, m'_i), (\text{TAG}'_j, m'_j)\}$ prevents trivial “transitivity” attacks where the adversary calls the oracle about the same (TAG_i, m_i, r_i) twice, but with different m'_i, m'_j . Then the oracle’s answers collide, as they yield the same value $\text{CHash}(pk, \text{TAG}_i, m_i; r_i)$ individually.

In our construction we also need that the tags generated by the signer and the ones by the sanitizer look identical (from the outside) but are generated differently (and that this is provable to a judge). Otherwise a malicious signer would be able to claim that a sanitized message has been the original. We resolve this by letting the tags of the sanitizer be truly random, whereas the tags of the signer need to be created pseudorandomly (with a pseudorandom generator PRG mapping n -bit inputs to $2n$ -bit outputs). In addition, the seed for the pseudorandomly generated labels should be recoverable for the signer from the signature and the secret key, such that we use a pseudorandom function PRF (mapping n -bit inputs to n -bit outputs for n -bit keys) to derive the seed for PRG from a nonce NONCE, included in the signature.

Finally, we also need a regular signature scheme $\mathcal{S} = (\text{SKGen}, \text{SSign}, \text{SVf})$ being existentially unforgeable under adaptive chosen-message attacks. Below we let (a_1, a_2, \dots) be some encoding of bit strings a_1, a_2, \dots into $\{0, 1\}^*$ such that (in contrast to concatenation $a_1 || a_2 || \dots$) all individual components are recoverable:

Construction 1 (Sanitizable Signature Scheme). Define the following sanitizable signature scheme $\text{SanSig} = (\text{KGen}_{\text{sig}}, \text{KGen}_{\text{san}}, \text{Sign}, \text{Sanit}, \text{Verify}, \text{Proof}, \text{Judge})$:

KEY GENERATION. Algorithm KGen_{sig} on input 1^n generates a key pair $(pk, sk) \leftarrow \text{SKGen}(1^n)$ of the underlying signature scheme, picks a key $\kappa \leftarrow \{0, 1\}^n$ for the pseudorandom function and returns $(pk_{\text{sig}}, sk_{\text{sig}}) = (pk, (sk, \kappa))$. Algorithm KGen_{san} for input 1^n returns a pair $(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \text{CHKGen}(1^n)$ of the chameleon hash scheme.

SIGNING. Algorithm Sign on input $m \in \{0, 1\}^{\ell}$, $sk_{\text{sig}}, pk_{\text{san}}, \text{ADM}$ picks $\text{NONCE} \leftarrow \{0, 1\}^n$ at random, computes $x = \text{PRF}(\kappa, \text{NONCE})$ and $\text{TAG} = \text{PRG}(x)$, and picks $r[j]$ for each j in ADM at random. It computes

$$h[j] = \begin{cases} \text{CHash}(pk_{\text{san}}, \text{TAG}, (j, m[j], pk_{\text{sig}}); r[j]) & \text{if } j \text{ is in ADM} \\ m[j] & \text{else} \end{cases}$$

for each block $m[j] \in \{0, 1\}^t$ and $\sigma_0 \leftarrow \text{SSign}(sk_{\text{sig}}, (h, pk_{\text{san}}, \text{ADM}))$ for $h = (h[1], h[2], \dots, h[\ell])$. It returns $\sigma = (\sigma_0, \text{TAG}, \text{NONCE}, \text{ADM}, r[j_1], \dots, r[j_k])$ where each j_i is in ADM .

SANITIZING. Algorithm Sanit on input a message m , information MOD , a signature $\sigma = (\sigma_0, \text{TAG}, \text{NONCE}, \text{ADM}, r[j_1], \dots, r[j_k])$, pk_{sig} and sk_{san} first checks that each modification in MOD is admissible according to ADM and that σ_0 is a valid signature for $(h, pk_{\text{san}}, \text{ADM})$. If not, it stops with output \perp . Else, for each j in ADM it lets $m'[j]$ be the modified block of $m[j]$ (possibly $m'[j] = m[j]$), picks new values $\text{NONCE}' \leftarrow \{0, 1\}^n$ and $\text{TAG}' \leftarrow \{0, 1\}^{2n}$ and replaces each $r[j]$ in the signature by

$$r'[j] \leftarrow \text{CHAdapt}(sk_{\text{san}}, \text{TAG}, (j, m[j], pk_{\text{sig}}), r[j], \text{TAG}', (j, m'[j], pk_{\text{sig}})).$$

It outputs m' and $\sigma' = (\sigma_0, \text{TAG}', \text{NONCE}', \text{ADM}, r'[j_1], \dots, r'[j_k])$.

VERIFICATION. Algorithm Verify on input a message $m \in \{0, 1\}^{\ell}$ and a signature $\sigma = (\sigma_0, \text{TAG}, \text{NONCE}, \text{ADM}, r[i_1], \dots, r[i_k])$, pk_{sig} and pk_{san} computes

$$h[j] = \begin{cases} \text{CHash}(pk_{\text{san}}, \text{TAG}, (j, m[j], pk_{\text{sig}}); r[j]) & \text{if } j \text{ is in ADM} \\ m[j] & \text{else} \end{cases}$$

and then outputs $\text{SVF}(pk_{\text{san}}, (h, pk_{\text{san}}, \text{ADM}), \sigma_0)$ for $h = (h[1], \dots, h[\ell])$.

PROOF. Algorithm Proof on input $sk_{\text{sig}}, m, \sigma$ and a sequence (m_i, σ_i) as well as pk_{san} searches the sequence to find a tuple $(\text{TAG}_i, (j, m_i[j], pk_{\text{sig}}), r[j])$ such that

$$\begin{aligned} & \text{CHash}(pk_{\text{san}}, \text{TAG}_i, (j, m_i[j], pk_{\text{sig}}), r_i[j]) \\ &= \text{CHash}(pk_{\text{san}}, \text{TAG}, (j, m[j], pk_{\text{sig}}), r[j]) \end{aligned}$$

for some distinct pair $(\text{TAG}, (j, m[j], pk_{\text{sig}}))$ in m, σ and where $\text{TAG}_i = \text{PRG}(x_i)$ for $x_i = \text{PRF}(\kappa, \text{NONCE}_i)$ for the value NONCE_i in σ_i . If it finds such data it returns this colliding tuple together with x_i , i.e.,

$$\pi = (\text{TAG}_i, (j, m_i[j], pk_{\text{sig}}), r_i[j], x_i),$$

else it outputs \perp .

JUDGE. The judge on input $m, \sigma, pk_{sig}, pk_{san}$ and $\pi = (\text{TAG}_\pi, (j, m_\pi[j], pk_{sig, \pi}), r_\pi[j], x_\pi)$ checks that $pk_{sig} = pk_{sig, \pi}$, that π describes a non-trivial collision under $\text{CHash}(pk_{san}, \cdot, \cdot, \cdot)$ for the pair $(\text{TAG}, j, m[j], pk_{sig}, r[j])$ in σ , i.e.,

$$\begin{aligned} & \text{CHash}(pk_{san}, \text{TAG}_\pi, (j, m_\pi[j], pk_{sig, \pi}); r_\pi[j]) \\ &= \text{CHash}(pk_{san}, \text{TAG}, (j, m[j], pk_{sig}); r[j]), \end{aligned}$$

that the block j is admissible, and that $\text{TAG}_\pi = \text{PRG}(x_\pi)$ for the given value x_π in the proof. If so, it outputs **San**, else it returns **Sig**.

Completeness of signatures generated by the signer follows easily from the completeness of the underlying signature scheme, completeness of signatures generated by the sanitizer follows from the fact that algorithm **CHAdapt** always returns a collision, and completeness for proofs holds as one always finds convincing data then.

5.2 Security

It remains to prove security:

Theorem 2. *The sanitizable signature scheme in Construction 1 is secure, i.e., it is immutable, transparent, sanitizer- and signer-accountable (and thus private and unforgeable), assuming that the chameleon hash function is collision-resistant under random-tagging attacks, that PRG and PRF are pseudorandom and that the signature scheme is existentially unforgeable under adaptive chosen-message attacks.*

Proof. We stepwise go through the properties. Most times we merely outline the security proof because a formalization is straightforward.

Immutability. Assume that the scheme is not immutable according to our definition and that there exists a successful adversary \mathcal{A} against this property. We show that this contradicts the unforgeability of the underlying signature scheme. There are two cases: Assume that \mathcal{A} succeeds by outputting $(pk_{san}^*, m^*, \sigma^*)$ such that $(pk_{san}^*, \text{ADM}^*, h^*)$ is different from all other data $(pk_{san, i}, \text{ADM}_i, h_i)$ appearing in the attack. Then the valid signature σ_0^* included in σ^* is for a message $(h^*, pk_{san}^*, \text{ADM}^*)$ which has not been signed with the underlying signature scheme before. This, however, contradicts the unforgeability of this signature scheme (observing that we can simulate **Proof** perfectly without knowledge of the secret key of the signature scheme).

Next assume $(pk_{san}^*, \text{ADM}^*, h^*)$ is identical to some $(pk_{san, i}, \text{ADM}_i, h_i)$. Then, since $pk_{san}^* = pk_{san, i}$ the messages m^* and m_i must differ in at least one inadmissible block j_i according to ADM_i . But since $\text{ADM}^* = \text{ADM}_i$ this must also be an inadmissible block according to ADM^* in m^* . Therefore $h^*[j_i] = m^*[j_i]$ must be different from $h_i[j_i] = m_i[j_i]$, contradicting the fact $h^* = h_i$. Hence, the second case cannot occur and the scheme is immutable.

Transparency. Transparency holds because with overwhelming probability all values NONCE picked by the signer are distinct and thus all x -values are computationally indistinguishable from independent and randomly chosen values. In this case all the generator's outputs, too, are indistinguishable from random $2n$ -bit strings (as chosen by the sanitizer). Given this the claim now follows from the distributional property of CHAdapt, that the sanitizing process goes through all admissible block and updates them, and the fact that the distribution of the input $(h, pk_{\text{san}}, \text{ADM})$ to the signing step is independent of the message. Hence, the distribution of the reply is computationally indistinguishable in the two cases for the Sanit/Sign box, independently of further queries to the signature, sanitizing or proof oracles (using the fact that the guessing the NONCE values in the signatures computed internally in the Sanit/Sign box is infeasible).

Sanitizer-Accountability. Assume that the scheme was not sanitizer-accountable and there was a successful adversary \mathcal{A} , i.e., such that Proof algorithm cannot find a non-trivial collision in the chameleon hashes for $(pk_{\text{san}}^*, m^*, \sigma^*)$ and the $(pk_{\text{san},i}^*, m_i, \sigma_i)$ queries. First note that if $(h^*, pk_{\text{san}}^*, \text{ADM}^*) \neq (h_i, pk_{\text{san},i}^*, \text{ADM}_i)$ for all i , the valid signature σ_0^* in σ^* for this tuple would constitute a successful forgery against the signature scheme (using again the fact that Proof can be easily simulated without the secret signing key).

Hence, there must be some i with $(h^*, pk_{\text{san}}^*, \text{ADM}^*) = (h_i, pk_{\text{san},i}^*, \text{ADM}_i)$. In particular, since a success requires $(pk_{\text{san}}^*, m^*) \neq (pk_{\text{san},i}^*, m_i)$ we must have $m^*[j] \neq m_i[j]$ for some block j . Furthermore, because $\text{ADM}^* = \text{ADM}_i$ and inadmissible message blocks are output in clear and cannot be distinct, it holds that

$$\begin{aligned} h^*[j] &= \text{CHash}(pk_{\text{san}}^*, \text{TAG}^*, (j, m^*[j], pk_{\text{sig}}^*); r^*[j]) \\ &= \text{CHash}(pk_{\text{san}}^*, \text{TAG}_i, (j, m_i[j], pk_{\text{sig}}^*); r_i[j]) = h_i[j] \end{aligned}$$

for some $r^*[j]$ in σ^* and $r_i[j]$ in σ_i . This, however, implies that Proof finds such a non-trivial collision with overwhelming probability. Given this, it is clear that Proof can also output x_i from the genuine signature data.

Signer-Accountability. We finally show signer-accountability, this time using the security under random-tagging attacks of the chameleon hash function. Assume that there is a successful attacker making the Judge accuse the sanitizer for a message which has not been sanitized by the legal sanitizer.

First note that for the adversary's successful output $pk_{\text{sig}}^*, m^*, \sigma^*$ (with tag TAG^*) and $\pi^* = (\text{TAG}_\pi, (j, m_\pi[j], pk_{\text{sig}\pi}, r_\pi[j], x_\pi)$ with overwhelming probability $\text{TAG}_\pi \neq \text{TAG}'_i$ for all i . This is so because with overwhelming probability no TAG'_i lies in the range of PRG and there cannot be a valid preimage x_π for $\text{TAG}_\pi = \text{TAG}'_i$. In particular, it follows that $\{\text{TAG}^*, \text{TAG}_\pi\} \neq \{\text{TAG}'_i, \text{TAG}'_j\}$ for all i, j .

Assume that $\{\text{TAG}^*, \text{TAG}_\pi\} \neq \{\text{TAG}_i, \text{TAG}'_i\}$ for all $i = 1, 2, \dots, q$. Then, because we also have $\{\text{TAG}^*, \text{TAG}_\pi\} \neq \{\text{TAG}'_i, \text{TAG}'_j\}$ this would straightforwardly contradict the security of the chameleon hash (noting that we can easily simulate

the sanitizer algorithm with the help of the OAdapt oracle). Hence, assume that $\{\text{TAG}^*, \text{TAG}_\pi\} = \{\text{TAG}_i, \text{TAG}'_i\}$ for some i and, since the random tags picked by the honest sanitizer are unique with overwhelming probability, we can assume that i is unique.

Because $\text{TAG}_\pi \neq \text{TAG}'_i$ we must have $\text{TAG}^* = \text{TAG}'_i$ and $\text{TAG}_\pi = \text{TAG}_i$. Since $(pk_{\text{sig}}^*, m^*) \neq (pk_{\text{sig},i}, m'_i)$ for a success there must be some j with $(\text{TAG}^*, (j, m^*[j], pk_{\text{sig}}^*)) \neq (\text{TAG}'_i, (j, m'_i[j], pk_{\text{sig},i}))$. However, assuming that all sanitizer tags are unique and observing that with overwhelming probability $\text{TAG}'_i \neq \text{TAG}_i$ and that for the same tag the prepended block numbers are distinct, it follows that the adversary has generated a new collision $(\text{TAG}^*, (j, m^*[j], pk_{\text{sig}}^*))$, $(\text{TAG}'_i, (j, m_\pi[j], pk_{\text{sig}}^*))$ which has not been queried previously. This would again contradict the security of the chameleon hash function and signer-accountability follows. \square

Acknowledgments

We thank the anonymous reviewers and the crypto group at Bristol for valuable comments. Marc Fischlin, Anja Lehmann and Dominique Schröder are supported by the Emmy Noether Programme Fi 940/2-1 of the German Research Foundation (DFG).

References

1. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: ESORICS. Volume 3679 of Lecture Notes in Computer Science., Springer (2005) 159–177
2. Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: ICISC. Volume 2288 of Lecture Notes in Computer Science., Springer (2001) 285–304
3. Miyazaki, K., Susaki, S., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H.: Digital documents sanitizing problem. In: Technical Report ISEC2003-20, IEICE (2003)
4. Johnson, R., Molnar, D., Song, D.X., Wagner, D.: Homomorphic signature schemes. In: CT-RSA. Volume 2271 of Lecture Notes in Computer Science., Springer (2002) 244–262
5. Miyazaki, K., Hanaoka, G., Imai, H.: Invisibly sanitizable digital signature scheme. IEICE Transactions **91-A**(1) (2008) 392–402
6. Klonowski, M., Lauks, A.: Extended sanitizable signatures. In: ICISC. Volume 4296 of Lecture Notes in Computer Science., Springer (2006) 343–355
7. Canard, S., Laguillaumie, F., Milhau, M.: Trapdoor sanitizable signatures and their application to content protection. In: ACNS. Lecture Notes in Computer Science, Springer-Verlag (2008) 258–276
8. Izu, T., Kanaya, N., Takenaka, M., Yoshioka, T.: Piats: A partially sanitizable signature scheme. In: ICICS. Volume 3783 of Lecture Notes in Computer Science., Springer (2005) 72–83
9. Miyazaki, K., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H., Tezuka, S., Imai, H.: Digitally signed document sanitizing scheme with disclosure condition control. IEICE Transactions **88-A**(1) (2005) 239–246

10. Izu, T., Kunihiro, N., Ohta, K., Takenaka, M., Yoshioka, T.: A sanitizable signature scheme with aggregation. In: ISPEC. Volume 4464 of Lecture Notes in Computer Science., Springer (2007) 51–64
11. Haber, S., Hatano, Y., Honda, Y., Horne, W., Miyazaki, K., Sander, T., Tezoku, S., Yao, D.: Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In: ASIACCS, ACM (2008) 353–362
12. Suzuki, M., Isshiki, T., Tanaka, K.: Sanitizable signature with secret information. In: In Proceedings of the Symposium on Cryptography and Information Security. (2006)
13. Yuen, T.H., Susilo, W., Liu, J.K., Mu, Y.: Sanitizable signatures revisited. In: CANS. Volume 5339 of Lecture Notes in Computer Science., Springer (2008) 80–97
14. Ateniese, G., de Medeiros, B.: On the key exposure problem in chameleon hashes. In: SCN. Volume 3352 of Lecture Notes in Computer Science., Springer (2004) 165–179

A General Message Modifications

In this section we outline how to adapt our security notions for more general message modifications. To this end we assume that ADM and MOD are (descriptions of) efficient algorithms such that $\text{ADM}(\text{MOD}) \in \{0, 1\}$ indicates if the modification is admissible and matches ADM , i.e., $\text{ADM}(\text{MOD}) = 1$. The function MOD maps any message m to the modified message $m' = \text{MOD}(m)$.

The notion of unforgeability remains unchanged. For immutability we demand as before that the adversary’s output $(pk_{\text{san}}^*, m^*, \sigma^*)$ describes a valid message-signature pair under keys $pk_{\text{sig}}, pk_{\text{san}}^*$. With the general message modification we now require for all queries to the signing oracle for $i = 1, 2, \dots, q$ that $pk_{\text{san}}^* \neq pk_{\text{san},i}$ or $m^* \notin \{\text{MOD}(m_i) \mid \text{MOD with } \text{ADM}(\text{MOD}) = 1\}$. Note that, under this general definition, it may not be efficiently verifiable if the adversary has succeeded.

The notion of privacy under general modifications demands that for each pair $(m_{j,0}, \text{MOD}_{j,0}, \text{ADM}_j), (m_{j,1}, \text{MOD}_{j,1}, \text{ADM}_j)$ submitted to the left-or-right oracle the modifications are admissible and yield the same message, i.e., we simply adapt the notation $(m_{j,0}, \text{MOD}_{j,0}, \text{ADM}_j) \equiv (m_{j,1}, \text{MOD}_{j,1}, \text{ADM}_j)$ accordingly. Transparency and the accountability notions remain unchanged.

We note that both security implications (transparency implies privacy and accountability implies unforgeability) are also valid under this more general notion. The separations remain true as block-based descriptions of MOD and ADM constitute a special case.