Unlinkability of Sanitizable Signatures

Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder

Darmstadt University of Technology, Germany
www.minicrypt.de

Abstract. Sanitizable signatures allow a designated party, called the sanitizer, to modify parts of signed data such that the immutable parts can still be verified with respect to the original signer. Ateniese et al. (ESORICS 2005) discuss five security properties for such signature schemes: unforgeability, immutability, privacy, transparency and accountability. These notions have been formalized in a recent work by Brzuska et al. (PKC 2009), discussing also the relationships among the security notions. In addition, they prove a modification of the scheme of Ateniese et al. to be secure according to these notions.
Here we discuss that a sixth property of sanitizable signature schemes may be desirable: unlinkability. Basically, this property prevents that one can link sanitized message-signature pairs of the same document, thus allowing to deduce combined information about the original document. We show that this notion implies privacy, the inability to recover the original data of sanitized parts, but is not implied by any of the other five notions. We also discuss a scheme based on group signatures meeting all six security properties.

1 Introduction

For a regular signature scheme any modification of the message makes the signature for the modified message invalid. In some applications, though, it may be preferable to support message modifications such that one can still verify the authenticity of the immutable message part, and that only authorized parties can make such changes. Signature schemes having this property are called *sanitizable*, as introduced by Ateniese et al. [1]. Related concepts have been discussed concurrently in [24,23,20].

Ateniese et al. [1] discuss the applicability of sanitizable signatures to anonymization of medical data, replacing commercials in authenticated media streams or updates of reliable routing information. They identified five desirable security properties for sanitizable signature schemes. Informally, these are:

Unforgeability. Says that no one except for the honest signer and sanitizer can create valid signatures.

Immutability. Demands that even a malicious sanitizer cannot change message parts which have not been marked as modifiable by the signer.

Privacy. Prevents an outsider to recover the original data of sanitized message parts.

Transparency. Covers the indistinguishability of signatures created by the signer or the sanitizer.

Accountability. Refers to the inability of a malicious signer or sanitizer to deny authorship.

Brzuska et al. [3] define these five properties with game-based approaches formally and relate them, showing that accountability implies unforgeability and transparency implies privacy; all other properties are independent. They also prove a modification of the scheme by Ateniese et al. [1] to be secure according to these five properties.

Unlinkability. Here we discuss that an additional property may be useful in some settings. We call this property *unlinkability* and motivate it by the following example (see also Figure 1): Assume that we have signed medical records and at some point we anonymize the data by redacting the personal information of the patients like names, addresses etc. At some other time, say for revenues reasons, we remove the actual medical treatments and leave only the personal information. Then one should not be able to link these data through the (sanitized) signatures and therefore reconstruct the full records. However, previous schemes like the one by Brzuska et al. [3] and, for example, the ones in [21,11,10] in fact allow such attacks. They are usually based on chameleon hashes which remain unchanged for the sanitization step and thus allow to identify two sanitized signatures derived from the same signature through the hash value. Other constructions like the one in [23] even come with an explicit document identifier, allowing to link sanitized messages easily.

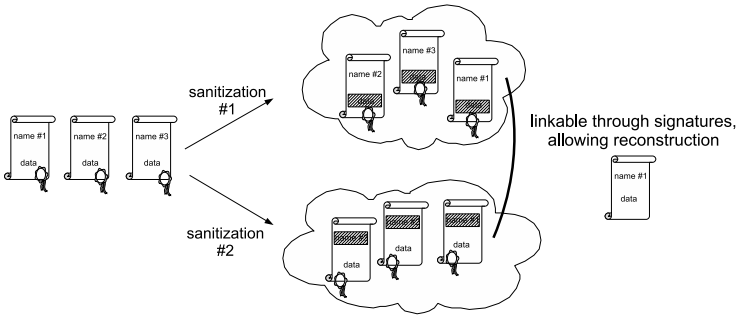

Fig. 1: Linkability problem

We hence introduce a formal definition of unlinkability and relate it to the previously given notions. It turns out that unlinkability is not implied by any of the other properties, but vice versa implies privacy. The reason is that privacy prevents an adversary of recovering the original data for sanitized parts, and violation of this property also enables the adversary to reconstruct and to link messages easily.

Construction. We then present a construction of a sanitizable signature scheme obeying all six properties, including unlinkability. The idea is fundamentally different from previous approaches which usually rely on chameleon hashes. In our case the signer first signs the fixed parts with a regular signature scheme. For the modifiable parts the signer and the sanitizer use a group signature scheme [13], i.e., a signature scheme which allows to sign anonymously on behalf of the group but such that a group manager can revoke the identity of the user that has signed [5]. In our case the group only consists of the signer and sanitizer, and the signer also incarnates the group manager. If the sanitizer later changes (some of) the modifiable message parts it can create a new group signature and replace the signer's group signature.

The anonymity of the group signature scheme in our context guarantees transparency (the indistinguishability of signatures originating from the signer and the sanitizer). The possibility to identify a group member by the group manager (i.e., the signer in our case) supports sanitizer-accountability, i.e., the ability to provide a proof that the sanitizer has created the signature. Signer-accountability is provided by the non-frameability of the group signature scheme which prevents a malicious group manager (i.e., the signer) from falsely accusing the sanitizer to be the source of a signature. Immutability follows from the unforgeability of the regular signature scheme for the fixed parts, and unlinkability from the fact that the sanitizer signs the entire message from scratch (the signature for fixed message parts remains unchanged).

We remark that the actual construction needs a careful implementation of the idea above to make the derived sanitizable signature scheme satisfy all desired security properties. This is in particular true since proposed group signature schemes in the literature like [5,9,22,14,18,19] come with varying security features and set-up assumptions. In this version we thus present a simple but not necessarily the most practical approach to turn our idea into a secure sanitizable scheme, e.g., following the definitions in [3] we do not rely on the fact that public keys of the signer or sanitizer are registered, although this is most likely in practice. In the full version we discuss further variations, e.g., multiple sanitizers, or using a ring signature scheme instead of a group signature scheme, thus dropping the accountability requirement for the derived sanitizable scheme.

Our solution shows that, in general, sanitizable signatures can be built from group signatures, thereby providing a new application for the latter primitive. This relation also immediately gives a feasibility result for sanitizable signatures: Since the work by Bellare et al. [5] about group signatures proves that one can derive them from IND-CCA secure encryption, non-interactive zero-knowledge proofs and digital signatures, all known to exist given trapdoor permutations, it follows that one can also build secure sanitizable signatures from trapdoor permutations.

Organization. In Section 2 we introduce the notion of sanitizable signatures and the security properties given in [1,3]. In Section 3 we discuss the notion of unlinkability and its relationship to the other security properties. In Section 4 we

present our construction of a secure sanitizable scheme based on group signatures.

2 Preliminaries

In this section we revisit the notion of sanitizable signatures and the previously given security properties.

2.1 Sanitizable Signatures

In a sanitizable signature scheme both the signer and the sanitizer hold a key pair $(sk_{\text{sig}}, pk_{\text{sig}})$, $(sk_{\text{san}}, pk_{\text{san}})$ such that the signer can sign messages with its secret key sk_{sig} and "attach" a description of the admissible modifications ADM which are allowed to the sanitizer pk_{san}. The sanitizer can then later change such a message according to some modification MOD and update the signature using his secret key sk_{san}. In order to settle disputes about the origin of a message-signature pair the algorithm Proof enables the signer to produce a proof π from previously signed messages that a signature has been created by the sanitizer. This proof can then be verified with the help of the Judge algorithm (but which only needs to decide about the origin in case of a valid message-signature pair in question; for invalid pairs such decisions are in general impossible).

To model admissible modifications we assume that ADM and MOD are (descriptions of) efficient deterministic algorithms such that MOD maps any message m to the modified message $m' = \text{MOD}(m)$, and $\text{ADM}(\text{MOD}) \in \{0,1\}$ indicates if the modification is admissible and matches ADM, in which case $\text{ADM}(\text{MOD}) = 1$. For example, for messages $m = m[1] \dots m[k]$ divided into blocks $m[i]$ of equal bit length t we can let ADM contain t and the indices of the modifiable blocks, and MOD then essentially consists of pairs $(j, m'[j])$ defining the new value for the j-th block.

For ease of notation we let FIX_{ADM} be an efficient deterministic algorithm which is uniquely determined by ADM and which maps m to the immutable message part $\text{FIX}_{\text{ADM}}(m)$, e.g., for block-divided messages $\text{FIX}_{\text{ADM}}(m)$ is the concatenation of all blocks not appearing in ADM. To exclude trivial examples we demand that admissible modifications leave the fixed part of a message unchanged, i.e., $\text{FIX}_{\text{ADM}}(m) = \text{FIX}_{\text{ADM}}(\text{MOD}(m))$ for all $m \in \{0,1\}^*$ and all MOD with $\text{ADM}(\text{MOD}) = 1$. Analogously, to avoid choices like FIX_{ADM} having empty output, we also require that the fixed part must be "maximal" given ADM, i.e., $\text{FIX}_{\text{ADM}}(m') \neq \text{FIX}_{\text{ADM}}(m)$ for $m' \notin \{\text{MOD}(m) \mid \text{MOD with } \text{ADM}(\text{MOD}) = 1\}$.

Jumping ahead, we note that for our construction based on group signatures we make another assumption on ADM. This property, denoted modification-decidability, allows to decide efficiently for given messages m, m^* and ADM whether m^* is an admissible modification of m with respect to ADM or not. This property is for example satisfied for the block-based approach. However, for our definitions of the security properties and their relationships we do not impose any restriction at this point.

The following definition is taken from [3]:

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

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

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 *of the admissibly modifiable message parts. It outputs a signature (or* $\perp$*, indicating an error):*

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

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 *and determines a new signature* σ' *for the modified message* $m' = \mathrm{MOD}(m)$*. Then* Sanit *outputs* m' *and* σ' *(or possibly* $\perp$ *in case of an error).*

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

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

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

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

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

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* $d \in \{\mathtt{Sig}, \mathtt{San}\}$ *indicating whether the message-signature pair has been created by the signer or the sanitizer:*

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

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. For a formal approach to correctness see [3].

2.2 Security of Sanitizable Signatures

Here we recall the security notions for sanitizable signatures given by Brzuska et al. [3]. We note that, there, they show that signer and sanitizer accountability together imply unforgeability, and that transparency implies privacy. Hence, in principle it suffices to show immutability, accountability and transparency. We therefore omit the formal definitions of unforgeability and privacy here and refer the reader to the full version of the paper.

Immutability. This property demands informally that a malicious sanitizer cannot change inadmissible blocks. 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 requiring that for each query $pk^*_{\text{san}} \neq pk_{\text{san},i}$ or $m^* \notin \{\text{MOD}(m_i) \mid \text{MOD with } \text{ADM}_i(\text{MOD}) = 1\}$ for the value ADM_i in σ_i, e.g., that for block-divided messages m^* and m_i differ in at least one inadmissible block. As the adversary can query the signer for several sanitizer keys pk_{san}, the security definition also covers the case where the signer interacts with several sanitizers simultaneously.

Definition 2 (Immutability). *A sanitizable signature scheme* SanSig *is* immutable *if for any efficient algorithm $\mathcal{A}$ the probability that the following experiment* $\mathsf{Immutability}^{\mathsf{SanSig}}_{\mathcal{A}}(n)$ *returns* 1 *is negligible (as a function of n).*

Experiment $\mathsf{Immutability}^{\mathsf{SanSig}}_{\mathcal{A}}(n)$
$(pk_{sig}, sk_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^n)$
$(pk^*_{san}, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{sig}, \cdot, \cdot), \mathsf{Proof}(sk_{sig}, \ldots, \cdot)}(pk_{sig})$
letting $(m_i, \text{ADM}_i, pk_{san,i})$ *and* σ_i *for* $i = 1, 2, \ldots, q$
denote the queries and answers to and from oracle Sign.
return 1 *if*
$\mathsf{Verify}(m^*, \sigma^*, pk_{sig}, pk^*_{san}) = \texttt{true}$ *and*
for all $i = 1, 2, \ldots, q$ *we have*
$pk^*_{san} \neq pk_{san,i}$ *or*
$m^* \notin \{\text{MOD}(m_i) \mid \text{MOD with } \text{ADM}_i(\text{MOD}) = 1\}$

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

In the sanitizer-accountability game let $\mathcal{A}_{\mathsf{Sanit}}$ be an efficient adversary playing the role of the malicious sanitizer. Adversary $\mathcal{A}_{\mathsf{Sanit}}$ has access to a Sign and Proof oracle. Her task is to output a valid message-signature pair m^*, σ^* together

with a key pk^*_{san} (with (pk^*_{san}, m^*) being different from pairs $(m_i, pksani)$ previously queried to the Sign oracle) such that the proof produced by the signer via Proof still leads the judge to decide "Sig", i.e., that the signature has been created by the signer.

Definition 3 (Sanitizer-Accountability). *One calls a sanitizable signature scheme* SanSig sanitizer-accountable *if for any efficient* $\mathcal{A}_{\mathsf{Sanit}}$ *the probability that the following experiment* $\mathsf{San\text{-}Accountability}^{\mathsf{SanSig}}_{\mathcal{A}_{\mathsf{Sanit}}}(n)$ *returns* 1 *is negligible (as a function of* n*).*

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

In the signer-accountability game a malicious signer $\mathcal{A}_{\mathsf{Sign}}$ gets a public sanitizing key pk_{san} as input. She 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 $\mathcal{A}_{\mathsf{Sign}}$ finally outputs a tuple $(pk^*_{\text{sig}}, m^*, \sigma^*)$ and is considered to succeed if Judge accuses the sanitizer for the new key-message pair pk^*_{sig}, m^* with a valid signature σ^*.

Definition 4 (Signer-Accountability). *A sanitizable signature scheme* SanSig *is called* signer-accountable *if for any efficient* $\mathcal{A}_{\mathsf{Sign}}$ *the probability that the following experiment* $\mathsf{Sig\text{-}Accountability}^{\mathsf{SanSig}}_{\mathcal{A}_{\mathsf{Sign}}}(n)$ *returns* 1 *is negligible (as a function of* n*):*

Experiment $\mathsf{Sig\text{-}Accountability}^{\mathsf{SanSig}}_{\mathcal{A}_{\mathsf{Sign}}}(n)$
$(pk_{san}, sk_{san}) \leftarrow \mathsf{KGen}_{san}(1^n)$
$(pk^*_{sig}, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sanit}(\cdot, \cdot, \cdot, \cdot, sk_{san})}_{\mathsf{Sign}}(pk_{san})$
letting (m'_i, σ'_i) *for* $i = 1, 2, \ldots, q$
denote the answers from oracle Sanit.
return 1 *iff*
$(pk^*_{sig}, m^*) \neq (pk_{sig,i}, m'_i)$ *for all* $i = 1, 2, \ldots, q$, *and*
$\mathsf{Verify}(m^*, \sigma^*, pk^*_{sig}, pk_{san}) = \texttt{true}$ *and*
$\mathsf{Judge}(m^*, \sigma^*, pk^*_{sig}, pk_{san}, \pi^*) = \texttt{San}$

Transparency. 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 algorithm to create $\sigma \leftarrow \mathsf{Sign}(m, sk_{\text{sig}}, pk_{\text{sig}}, \text{ADM})$, then runs the sanitizer algorithm 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 $\frac{1}{2}$. Below we also define a relaxed version called *proof-restricted transparency* and discuss the idea after the definition.

Definition 5 ((Proof-Restricted) Transparency). *A sanitizable signature scheme* SanSig *is* (proof-restricted) transparent *if for any efficient algorithm* $\mathcal{A}$ *the probability that the following experiment* $\mathsf{Transparency}_{\mathcal{A}}^{\mathsf{SanSig}}(n)$ *returns* 1 *is negligibly close to* $\frac{1}{2}$.

Experiment $\mathsf{Transparency}_{\mathcal{A}}^{\mathsf{SanSig}}(n)$
$(pk_{sig}, sk_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^n)$
$(pk_{san}, sk_{san}) \leftarrow \mathsf{KGen}_{san}(1^n)$
$b \leftarrow \{0, 1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{sig}, \cdot, \cdot), \mathsf{Sanit}(\cdot, \cdot, \cdot, \cdot, sk_{san}), \mathsf{Proof}(sk_{sig}, \ldots, \cdot), \mathsf{Sanit/Sign}(\cdot, \cdot, \cdot, sk_{sig}, sk_{san}, pk_{sig}, pk_{san}, b)}$
with input (pk_{sig}, pk_{san})
where oracle $\mathsf{Sanit/Sign}$ *for input* $m_k, \text{MOD}_k, \text{ADM}_k$
first computes $\sigma_k \leftarrow \mathsf{Sign}(m_k, sk_{sig}, pk_{san}, \text{ADM}_k)$,
then computes $(m'_k, \sigma'_k) \leftarrow \mathsf{Sanit}(m_k, \text{MOD}_k, \sigma_k, pk_{sig}, sk_{san})$,
then, if $b = 1$, *replaces* σ'_k *by* $\sigma'_k \leftarrow \mathsf{Sign}(m'_k, sk_{sig}, pk_{san}, \text{ADM}_k)$,
and finally returns (m'_k, σ'_k).
return 1 *iff*
$a = b$
(and, in the proof-restricted case, $\mathcal{A}$ *has not queried*
any m'_k *output by* $\mathsf{Sanit/Sign}$ *to* Proof*)*

The original definition of Brzuska et al. [3] does not consider the proof-restricted case. Without this restriction, though, achieving transparency at first seems to be impossible because the adversary can then always submit the replies of the Sanit/Sign oracle to the Proof oracle and thereby recover the secret bit b. However, in their construction the Proof algorithm searches in the list of previously signed messages and only gives a useful answer if it finds a match, enabling transparency without this restriction. Yet, any solution (like ours here) where the Proof algorithm is "history-free" can only achieve the proof-restricted version. Note that Proof algorithms forgoing the set of previously signed messages are preferable from an efficiency point of view, of course.

As for the implications among the security notions [3] we note that proof-restricted transparency only implies a proof-restricted form of privacy, where the answer messages of the LoRSanit oracle cannot be submitted to the Proof oracle either. However, since we show in the next section that unlinkability implies full

privacy and our construction achieves unlinkability, our scheme is also private in the non-restricted sense. We note that all the separation results in [3] remain valid for proof-restricted transparency.

3 Unlinkability

In this section we define unlinkability formally and discuss its relationship to the other security notions.

3.1 Definition

As explained in the introduction, unlinkability refers to the impossibility to use the signatures to identify sanitized message-signature pairs originating from the same source. Technically, we use an indistinguishability-based approach to define this property, saying that, given a signature for a sanitized message of two possible sources, the adversary cannot predict the actual original message better than by guessing. This should even hold if the adversary herself provides the two source message-signature pairs and modifications of which one is sanitized. The stipulation here is that the two modifications yield the same sanitized message. Else, if for example the sanitized messages still contain some unique but distinct entry, then predicting the source is easy, of course. This, however, is beyond the scope of signature schemes: the scheme should only prevent that *signatures* can be used to link data.

Formally, we use a game-based approach to define unlinkability, similar to the other security notions in [3]. The adversary is given access to a signing oracle and a sanitizer oracle (and a proof oracle since this step depends on the signer's secret key and may leak valuable information). The adversary is also allowed to query a left-or-right oracle $\mathsf{LoRSanit}$ which is initialized with a secret random bit b. In each of the multiple queries to $\mathsf{LoRSanit}$ the adversary provides a pair of tuples, each consisting of a message, a modification and a *valid* signature, such that the recoverable description of admissible modifications is identical in both cases (since we assume that ADM is recoverable from a signature providing distinct descriptions ADM would allow a trivial attack; so would the case that only one signature is valid). Depending on the bit b, the adversary gets the sanitized message-signature pair of either the left or right input pair. The adversary should eventually predict the bit b significantly better than with the guessing probability of $\frac{1}{2}$.

Definition 6 (Unlinkability). *A sanitizable signature scheme* SanSig *is* unlinkable *if for any efficient algorithm* $\mathcal{A}$ *the probability that the following experiment* $\mathsf{Unlinkability}^{\mathsf{SanSig}}_{\mathcal{A}}(n)$ *returns* 1 *is negligibly close to* $\frac{1}{2}$.

Experiment $\mathsf{Unlinkability}^{\mathsf{SanSig}}_{\mathcal{A}}(n)$
$(pk_{sig}, sk_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^n)$
$(pk_{san}, sk_{san}) \leftarrow \mathsf{KGen}_{san}(1^n)$
$b \leftarrow \{0,1\}$

$a \leftarrow \mathcal{A}^{\mathsf{Sign}(sk_{sig},\cdots),\mathsf{Sanit}(sk_{san},\cdots),\mathsf{Proof}(sk_{sig},\cdots),\mathsf{LoRSanit}(sk_{sig},sk_{san},b,\cdots)}(pk_{sig}, pk_{san})$

where oracle $\mathsf{LoRSanit}(\cdot,\cdot,\cdot,sk_{sig},sk_{san},b)$*, on input*
$(m_{j,0}, \mathrm{MOD}_{j,0}, \sigma_{j,0}, m_{j,1}, \mathrm{MOD}_{j,1}, \sigma_{j,1})$ *with recoverable* $\mathrm{ADM}_{j,0} = \mathrm{ADM}_{j,1}$
$\mathsf{Verify}(m_{j_0}, \sigma_{j,0}, pk_{sig}, pk_{san}) = \mathtt{true}$*,* $\mathsf{Verify}(m_{j_1}, \sigma_{j,1}, pk_{sig}, pk_{san}) = \mathtt{true}$*,*
returns $(m'_j, \sigma'_j) \leftarrow \mathsf{Sanit}(m_{j,b}, \mathrm{MOD}_{j,b}, \sigma_{j,b}, pk_{sig}, sk_{san})$*,*
and where $(m_{j,0}, \mathrm{MOD}_{j,0}, \mathrm{ADM}_{j,0}) \equiv (m_{j,1}, \mathrm{MOD}_{j,1}, \mathrm{ADM}_{j,1})$*,*
i.e., are mapped to the same modified message.

return 1 *if* $a = b$*.*

A pictorial description is given in Figure 2. We note that the definition above is for example robust concerning several sanitization steps in the LoRSanit oracle. That is, we could allow the adversary in the experiment above to submit arbitrarily long "modification chains" $\mathrm{MOD}^1_{j,0}, \ldots, \mathrm{MOD}^m_{j,0}$ and $\mathrm{MOD}^1_{j,1}, \ldots, \mathrm{MOD}^m_{j,1}$ such that the two source documents are gradually sanitized with a match in the resulting documents. Still, predicting b remains hard, as such chains can potentially be simulated by calling the sanitizer oracle for the first $m-1$ modifications manually, before entering the final sanitization step into the LoRSanit oracle.

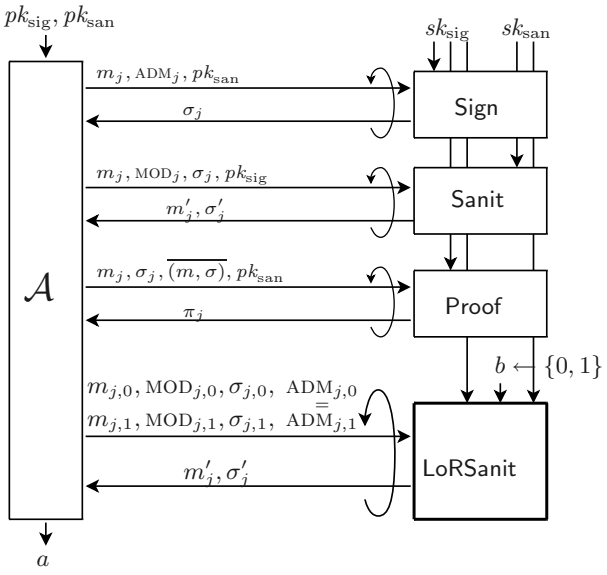

Fig. 2: Unlinkability. $\mathcal{A}$ wins if it outputs $a = b$.

Recall the example of medical records which are sanitized twice, one time basically removing the personal information and the other time removing the medical data. Our notion of unlinkability can then be used to show that such sanitized message-signature pairs do not allow to reconstruct the full data better than by guessing. Assume for simplicity that we only have two records with entries $(\texttt{name\#0}, \texttt{data\#0})$ and $(\texttt{name\#1}, \texttt{data\#1})$. Then we create all four possible

combinations ($\texttt{name\#}a, \texttt{data\#}b$) for $a, b \in \{0,1\}$ and ask for signatures for them (with both parts being admissibly changeable). For each $a \in \{0,1\}$ we then insert the pairs ($\texttt{name\#}a, \texttt{data\#0}$) and ($\texttt{name\#}a, \texttt{data\#1}$) twice into the LoRSanit oracle, one time cutting off the name-part, the other time removing the data-part. Altogether we make thus four calls to the LoRSanit oracle, and we hand those four replies to the adversary. Our unlinkability definition says that one cannot distinguish the two cases (left or right sanitization) better than by guessing, thus also disallowing to tell which data belong to whose name.

Our definition above is for unlinkability with respect to message-signature pairs sanitized by the *same* sanitizer. One can easily extend the above definition by demanding that the adversary can also determine different sanitizers for the left and for the right input data. But then both sanitizers must have been declared to have the permission to sanitize, otherwise one could easily determine the secret bit of the LoRSanit by picking an invalid sanitizer for one of the input tuples.

3.2 Relationships of the Security Notions

We first show that unlinkability does not follow from any of the other security requirements. Then we prove that unlinkability implies privacy, and finally discuss that unlinkability does not imply any of the other properties.

Proposition 1 (Independence of Unlinkability). *Assume that there exists a sanitizable signature scheme (obeying one or more of the properties unforgeability, immutability, privacy, (proof-restricted) transparency, signer-accountability and sanitizer-accountability). Then there exists a sanitizable signature scheme which is not unlinkable but preserves the other security properties.*

The proof follows by simply appending a unique identifier id to each signature. This does not destroy any of the other security properties but clearly violates unlinkability. The proof of the following is straightforward as the privacy experiment is essentially the unlinkability experiment with less control for the adversary:

Proposition 2 (Unlinkability Implies Privacy). *Any unlinkable sanitizable signature scheme is also private.*

With the next proposition we show that unlinkability does not imply any of the other security properties (assuming that we start with a secure sanitizable signature scheme like the one we construct in the next section):

Proposition 3 (Independence of Other Properties). *Assume that there exists a sanitizable signature scheme which is unforgeable, immutable, private, (proof-restricted) transparent, signer-accountable, sanitizer-accountable and unlinkable. Then for any of the properties immutability, transparency, unlinkability, signer-accountability and sanitizer-accountability, there exists a sanitizable signature scheme obeying all properties except for the one in question.*

Proof. The fact that unlinkability does not follow from the other properties has already been shown in Proposition 1. For the other properties we remark that the counterexamples in [3] which seperate immutability, transparency, signer-accountability and sanitizer-accountability from the other properties also preserve unlinkability in each case (and also hold for proof-restricted transparency). □

4 Constructions based on Group Signatures

In this section we present our unlinkable sanitizable signature scheme (which also satisfies the other security properties). As explained in the introduction, the idea is to use a group signature scheme for the group consisting of the signer and the sanitizer, such that the signer signs the immutable message part with a regular signature scheme and the full message with the group signature scheme. The sanitizer can then update the full message and only sign this second component. The signer also takes over the role of the group manager in order to provide accountability.

4.1 Group Signatures

Group signatures, introduced by Chaum and van Heyst [13], allow a set of users to sign on behalf of the group such that outsiders cannot distinguish between different signers (anonymity) but such that a group manager can trace the signer's identity (traceability). We follow the formal framework of Bellare et al. [5] but add the non-frameability requirement [9] that even the group manager cannot sign on behalf of the users. Recall that this is necessary for the accountability in our sanitizable signature scheme, where the signer acts as the group manager and should not be able to make the judge falsely accuse the sanitizer.

We briefly recall group signature schemes and their security properties. For comprehensive definitions see the full version of the paper and [5]. A group signature scheme GS consists of six efficient algorithms $\mathsf{GS} = (\mathsf{GKGen}, \mathsf{UKGen}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open}, \mathsf{GJudge})$ where

- $(sk_{\text{user}}, pk_{\text{user}}) \leftarrow \mathsf{UKGen}(1^n)$ generates individual user key pairs,
- $(gmsk, gpk, \mathbf{cert}) \leftarrow \mathsf{GKGen}(1^n, \mathbf{gpk}_{user})$ takes the tuple $\mathbf{gpk}_{user}$ of the users' public keys and generates a group manager secret key $gmsk$, a group public key gpk and an individual certificate $cert_i$ for each user, where $\mathbf{cert}$ designates the tuple of all $cert_i$,
- $\sigma \leftarrow \mathsf{GSig}(sk_{\text{user},i}, cert_i, gpk, m)$ signs a message m given the user's secret data $sk_{\text{user},i}$, $cert_i$ and the group's public key gpk,
- $(i, \pi) \leftarrow \mathsf{Open}(gmsk, m, \sigma, \mathbf{gpk}_{user}, gpk)$ on input a message m and signature σ returns the index i of the alleged signer and a proof π such that
- $\mathsf{GJudge}(m, \sigma, i, \pi, gpk, \mathbf{gpk}_{user})$ either confirms the accusation or denies it.

There are three security properties for group signatures [5,9]:

Anonymity. Means that one cannot tell from a group signature who signed a message, even if one knows the secret data of the user and can ask the group manager to reveal the identities for other signatures.

Traceability. Refers to the fact that a malicious user cannot falsely accuse an honest user to be the signer of a message, even if the malicious user is allowed to see other signatures generated by this honest user and can call the group manager.

Non-Frameability. Strengthens traceability in the sense that even if the malicious user colludes with the group manager they cannot frame an honest user.

Definition 7 (Secure Group Signature). *We call a group signature scheme* secure *if it is anonymous and non-frameable.*

Note that we tailor the group signature definitions to our needs thereby adding non-frameability, making the scheme syntax setup session free and relaxing the security model concerning some technical issues which are discussed in the full version of this paper. As for instantiations we remark that the (generic) construction by Bellare et al. [5] satisfies our adapted definitions. As for more efficient group signature schemes, we can implement our sanitizable signature scheme with other group signature schemes like [22,14,18,19]. Yet, these group signature schemes need additional set-up assumptions like a trusted party generating common parameters or interactive registration of users. Our sanitizable signature scheme then inherits these characteristics (recall that, in practice, registration of signer and sanitizer keys is for example necessary to provide meaningful accountability).

4.2 Construction

In this section we show that the new security requirement of unlinkability can be achieved in combination with the five established security properties formalized in [3]. Recall that we sign the entire message, including the modifiable parts, with the group signature scheme, and —in order to prevent inadmissible changes— the signer also signs the fixed part with a regular scheme. This requires some care because if we take an arbitrary signature scheme then the signature itself may act as a unique identifier, even for messages with identical fixed parts. Thereby, unlinkability would be violated.

The solution is to use a secure deterministic signature scheme for the fixed part (such that the signature is identical for messages with the same fixed part). Alternatively, one can deploy a rerandomizable signature scheme such that the sanitizer can rerandomize the signature, excising the link to the input signature. Below we use the "deterministic solution" for simplicity, and since every secure signature scheme can be easily turned into a deterministic one via pseudorandom functions [15].

For a formal definition of *strongly unforgeable* signature schemes see [17]. We need this unforgeability notion (saying that one cannot even find a new

signature for a previously signed message) to provide unlinkability. Examples of signature schemes achieving this strong notion are [6,12,4,2,8]. Moreover, it is possible to obtain a strongly unforgeable signature scheme out of any unforgeable signature scheme applying the transformation of Bellare and Shoup [7]. Applying the transformation of [15] one can then make such schemes also deterministic.

Recall that the idea behind our scheme is that for each signature the signer uses a group manager key, creates a certified user key to sign the modifiable parts, and certifies the sanitizer's public key as a group member to support modifications. But since our definition of sanitizable signatures demands state-free solutions, the signer formally cannot store the group manager key for this sanitizer and would need to create a new one for each call. We bypass this as follows: we let the signer for each signing request, including a public key of the sanitizer pk_{san}, create the group manager's keys etc. via the corresponding group signature algorithms, but provide the randomness for these algorithms by applying a pseudorandom function to pk_{san} (see [16] for a definition of pseudorandom functions). By this, we end up with (almost) independent keys for different sanitizers, but use consistent parameters for each sanitizer. For the same reason we also include the group membership certificate of the sanitizer in the signature, although it would be given directly to the sanitizer instead. As a side effect, since the group manager's public key is tied to the sanitizer in question, we also rely on group signatures with static joins only.

Construction 1 (Sanitizable Signature Scheme). *Let* $\mathcal{S} = (\mathsf{SKGen}, \mathsf{SSign}, \mathsf{SVf})$ *be a (regular) signature scheme, let* $\mathsf{GS} = (\mathsf{GKGen}, \mathsf{UKGen}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open}, \mathsf{GJudge})$ *be a group signature scheme. Let* $\mathcal{PRF} = (\mathsf{KGen}_{prf}, \mathsf{PRF})$ *be pseudorandom function. Define the sanitizable signature scheme* $\mathsf{SanSig} = (\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$ *as follows:*

Key Generation. *First, algorithm* KGen_{sig} *gets the input* 1^n *and runs* $(ssk, spk) \leftarrow \mathsf{SKGen}(1^n)$ *to create a key pair for the signature scheme, and then also invokes* $k \leftarrow \mathsf{KGen}_{prf}(1^n)$ *to derive a key for the pseudorandom function. It outputs* $(sk_{sig}, pk_{sig}) = ((ssk, k), spk)$. *Algorithm* $\mathsf{KGen}_{san}(1^n)$ *generates a key pair* $(sk_{san}, pk_{san}) = (gsk_{san}, gpk_{san}) \leftarrow \mathsf{UKGen}(1^n)$ *of the group signature scheme.*

Signing. *Algorithm* Sign *on input* $m \in \{0,1\}^*$, $sk_{sig} = (ssk, k)$, pk_{san}, ADM *sets* $m_{\text{FIX}} = \text{FIX}_{\text{ADM}}(m)$ *for the algorithm* FIX_{ADM} *determined by* ADM. *It runs the user key generation algorithm* $(gsk_{sig}, gpk_{sig}) \leftarrow \mathsf{UKGen}(1^n; \mathsf{PRF}(k, 0\|pk_{san}))$ *for randomness* $\mathsf{PRF}(k, 0\|pk_{san})$ *and afterwards the group manager algorithm to compute*

$$(gmsk, gpk, cert_{sig}, cert_{san}) \leftarrow \mathsf{GKGen}(1^n, (gpk_{sig}, pk_{san}); \mathsf{PRF}(k, 1\|pk_{san}))$$

for randomness $\mathsf{PRF}(k, 1\|pk_{san})$. *It computes*

$$\sigma_{\text{FIX}} = \mathsf{SSign}(ssk, (m_{\text{FIX}}, \text{ADM}, pk_{san}, gpk)) \text{ and}$$

$$\sigma_{\text{FULL}} = \mathsf{GSig}(gsk_{sig}, cert_{sig}, (m, pk_{sig}), gpk)$$

using the signing algorithms of the regular and of the group signature scheme. The algorithm finally returns $\sigma = (\sigma_{\text{FIX}}, \sigma_{\text{FULL}}, \text{ADM}, pk_{san}, cert_{san}, gpk)$.

SANITIZING. *Algorithm* Sanit *on input a message* m*, information* MOD*, a signature* $\sigma = (\sigma_{\text{FIX}}, \sigma_{\text{FULL}}, \text{ADM}, pk_{san}, cert_{san}, gpk)$*, keys* pk_{sig} *and* sk_{san} *first recovers* $m_{\text{FIX}} = \text{FIX}_{\text{ADM}}(m)$*. It then checks that* MOD *is admissible according to* ADM *and that* σ_{FIX} *is a valid signature for message* $(m_{\text{FIX}}, \text{ADM}, pk_{san}, gpk)$ *under key spk. If not, it stops outputting* $\perp$*. Else, it derives the modified message* $m' = \text{MOD}(m)$ *and computes*

$$\sigma'_{\text{FULL}} = \mathsf{GSig}(gsk_{san}, cert_{san}, (m', pk_{sig}), gpk)$$

and outputs m' *together with* $\sigma' = (\sigma_{\text{FIX}}, \sigma'_{\text{FULL}}, \text{ADM}, pk_{san}, cert_{san}, gpk)$.

VERIFICATION. *Algorithm* Verify *gets as input a message* $m \in \{0,1\}^*$*, a signature* $\sigma = (\sigma_{\text{FIX}}, \sigma_{\text{FULL}}, \text{ADM}, pk_{san}, cert_{san}, gpk)$ *and public keys* $pk_{sig} = spk$ *and* pk_{san}*. It first recovers* $m_{\text{FIX}} = \text{FIX}_{\text{ADM}}(m)$*. It then checks whether* $\mathsf{SVf}(spk, (m_{\text{FIX}}, \text{ADM}, pk_{san}, gpk), \sigma_{\text{FIX}}) = 1$ *and* $\mathsf{GVf}(gpk, (m, pk_{sig}), \sigma_{\text{FULL}})$ *verifies under the group public key as* `true`*, too. If so, it outputs* 1*, declaring the entire signature as valid. Otherwise it returns* 0*, indicating an invalid signature.*

PROOF. *Algorithm* Proof *gets as input* sk_{sig}*,* m *and* $\sigma = (\sigma_{\text{FIX}}, \sigma_{\text{FULL}}, \text{ADM}, pk_{san}, cert_{san}, gpk)$*. It parses the key as* $sk_{sig} = (ssk, k)$ *and recomputes*

$$(gmsk, gpk', cert'_{sig}, cert'_{san}) = \mathsf{GKGen}(1^n, (gpk_{sig}, pk_{san}); \mathsf{PRF}(k, 1||pk_{san}))$$

and checks that $gpk' = gpk$ *and* $cert'_{san} = cert_{san}$ *(and immediately returns* $\perp$ *if not). It next verifies that* $\mathsf{SVf}(spk, (m_{\text{FIX}}, \text{ADM}, pk_{san}, gpk), \sigma_{\text{FIX}}) = 1$ *and, if so, computes and outputs* $(i, \pi) \leftarrow \mathsf{Open}(gmsk, (m, pk_{sig}), \sigma_{\text{FULL}}, gpk)$*, where* $i \in \{\mathtt{Sig}, \mathtt{San}\}$ *is the identity returned by the* Open *algorithm (or,* Proof *returns* $\perp$ *if any of the verification steps above fail).*

JUDGE. *The judge on input* $m, \sigma, pk_{sig}, pk_{san}$ *and a proof* (i, π) *with* $i \in \{\mathtt{Sig}, \mathtt{San}\}$ *parses* σ *as* $(\sigma_{\text{FIX}}, \sigma_{\text{FULL}}, \text{ADM}, pk_{san}, cert_{san}, gpk)$*. It derives* $b \leftarrow \mathsf{GJudge}((m, pk_{sig}), \sigma_{\text{FULL}}, i, \pi, gpk)$ *using the judge algorithm of the group signature scheme. If* $b =$ `true` *it outputs* i*, else it outputs* $i = \mathtt{Sig}$*.*

Completeness of signatures generated by the signer and sanitizer follows easily from the completeness of the underlying signature schemes and the fact that FIX_{ADM} leaves the fixed message parts unchanged for modified messages. There is a negligible probability that a signature of the signer or the sanitizer also verifies under the other party's other key, yielding possibly a wrong answer from the judge. We ignore this issue here for simplicity.

4.3 Security Proof

We need an additional property of the admissible modifications ADM: given arbitrary messages $m, m^* \in \{0,1\}^*$ (and a security parameter 1^n) it should be efficiently decidable whether $m^* \in \{\text{MOD}(m) \mid \text{MOD with } \text{ADM}(\text{MOD}) = 1\}$ or not. We call such ADM *modification-decidable* and a sanitizable signature scheme

modification-restricted if it only allows modification-decidable ADM. As an example consider again block-divided messages where ADM describes the block-length and the indices of changeable blocks. Then it is easy to check whether m^* has been changed in admissible blocks only or not.

Theorem 2. *Let $\mathcal{S}$ be a strongly unforgeable deterministic signature scheme and let GS be a secure group signature scheme. Assume further that $\mathcal{PRF}$ is a pseudorandom function. Then the modification-restricted sanitizable signature scheme in Construction 1 is unforgeable, immutable, private, proof-restricted transparent, accountable and unlinkable.*

As unlinkability implies privacy, and as moreover, sanitizer-accountability and signer-accountability imply unforgeability, it suffices to prove these two types of accountability as well as with unlinkability, immutability and (proof-restricted) transparency.

For the proof idea note that we can reduce transparency of our sanitizable signatures to anonymity of the underlying group signature scheme. Traceability of the group signature scheme enables the group manager (i.e., the signer) to provide a proof that a message has indeed been signed by a certain group member. Thus, if the sanitizer signs a message, the signer can produce evidence that this signature originates from the sanitizer. This shows sanitizer-accountability. Vice versa, the unframeability property of group signature scheme assures that the group manager (i.e., the signer) cannot falsely accuse a group member of having signed a message. Therefore, signer-accountability follows from unframeability.

The unforgeability of the underlying regular signature scheme assures immutability: If the sanitizer changed admissible parts of a message, she would be obliged to forge a signature for the fixed part. Unlinkability holds as the sanitizer creates a new group signature from scratch when sanitizing. Furthermore, the signature of the regular signature scheme remains unchanged, and is identical for different documents with the same fixed part because we use a deterministic scheme. The formal proof follows these ideas and appears in the full paper.

Acknowledgments

We thank the anonymous reviewers for valuable comments. Marc Fischlin, Anja Lehmann and Dominique Schröder are supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG). This work was also supported by CASED (`www.cased.de`).

References

1. Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. *Sanitizable Signatures*. ESORICS, Volume 3679 of Lecture Notes in Computer Science, pages 159–177. Springer, 2005.
2. Dan Boneh and Xavier Boyen. *Short Signatures Without Random Oracles*. Advances in Cryptology — Eurocrypt'04, Volume 3027 of Lecture Notes in Computer Science, pages 56–73. Springer-Verlag, 2004.

3. Christina Brzuska, Marc Fischlin, Tobias Freudenreich, Anja Lehmann, Marcus Page, Jakob Schelbert, Dominique Schroeder, and Florian Volk. *Security of Sanitizable Signatures Revisited*. Public-Key Cryptography (PKC) 2009, Volume 5443 of Lecture Notes in Computer Science, pages 317–336. Springer-Verlag, 2009.
4. Dan Boneh, Ben Lynn, and Hovav Shacham. *Short Signatures from the Weil Pairing*. *Journal of Cryptology*, 17(4):297–319, 2004.
5. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. *Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions*. Advances in Cryptology — Eurocrypt2003, Volume 2656 of Lecture Notes in Computer Science, pages 614–629. Springer-Verlag, 2003.
6. Mihir Bellare and Phillip Rogaway. *The Exact Security of Digital Signatures - How to Sign with RSA and Rabin*. Advances in Cryptology — Eurocrypt 1996, pages 399–416. Springer-Verlag, 1996.
7. Mihir Bellare and Sarah Shoup. *Two-Tier Signatures, Strongly Unforgeable Signatures, and Fiat-Shamir Without Random Oracles*. Public-Key Cryptography (PKC)'07, Lecture Notes in Computer Science, pages 201–216. Springer-Verlag, 2007.
8. Dan Boneh, Emily Shen, and Brent Waters. *Strongly Unforgeable Signatures Based on Computational Diffie-Hellman*. Public-Key Cryptography (PKC)'06, Lecture Notes in Computer Science, pages 229–240. Springer-Verlag, 2006.
9. Mihir Bellare, Haixia Shi, and Chong Zhang. *Foundations of Group Signatures: The Case of Dynamic Groups*. CT-RSA 2005, Volume 3376 of Lecture Notes in Computer Science. Springer-Verlag, 2005.
10. Sébastien Canard and Amandine Jambert. *On Extended Sanitizable Signature Schemes*. Topics in Cryptology — Cryptographer's Track, RSA Conference (CT-RSA) 2010, Volume 5985 of Lecture Notes in Computer Science, pages 179–194. Springer-Verlag, 2010.
11. Sébastien Canard, Fabien Laguillaumie, and Michel Milhau. *Trapdoor Sanitizable Signatures and Their Application to Content Protection*. ACNS, Lecture Notes in Computer Science, pages 258–276. Springer-Verlag, 2008.
12. Jean-Sebastien Coron. *On the Exact Security of Full Domain Hash*. Advances in Cryptology — Crypto2000, Volume 1880 of Lecture Notes in Computer Science, pages 229–235. Springer-Verlag, 2000.
13. D. Chaum and E. van Heyst. *Group Signatures*. Advances in Cryptology — Eurocrypt'91, Volume 547 of Lecture Notes in Computer Science, pages 241–246. Springer-Verlag, 1991.
14. Cécile Delerablée and David Pointcheval. *Dynamic Fully Anonymous Short Group Signatures*. VIETCRYPT, Volume 4341 of Lecture Notes in Computer Science, pages 193–210. Springer-Verlag, 2006.
15. Oded Goldreich. *Two Remarks Concerning the Goldwasser-Micali-Rivest Signature Scheme*. Advances in Cryptology — Crypto '86, Volume 263 of Lecture Notes in Computer Science, pages 104–110. Springer-Verlag, 1987.
16. Oded Goldreich. *The Foundations of Cryptography*, Volume 1. Cambridge University Press, 2001.
17. Oded Goldreich. *The Foundations of Cryptography*, Volume 2. Cambridge University Press, 2004.
18. Jens Groth. *Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures*. Advances in Cryptology — Asiacrypt, Volume 4284 of Lecture Notes in Computer Science, pages 444–459. Springer-Verlag, 2006.

19. Jens Groth. *Fully Anonymous Group Signatures Without Random Oracles*. Advances in Cryptology — Asiacrypt, Volume 4833 of Lecture Notes in Computer Science, pages 164–180. Springer-Verlag, 2007.
20. Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. *Homomorphic Signature Schemes*. CT-RSA, Volume 2271 of Lecture Notes in Computer Science, pages 244–262. Springer, 2002.
21. Marek Klonowski and Anna Lauks. *Extended Sanitizable Signatures*. ICISC, Volume 4296 of Lecture Notes in Computer Science, pages 343–355. Springer, 2006.
22. Aggelos Kiayias and Moti Yung. *Group Signatures with Efficient Concurrent Join*. Advances in Cryptology — Eurocrypt 2005, Volume 3494 of Lecture Notes in Computer Science, pages 198–214. Springer-Verlag, 2005.
23. K. Miyazaki, S. Susaki, M. Iwamura, T. Matsumoto, R. Sasaki, and H. Yoshiura. *Digital documents sanitizing problem*. Technical Report ISEC2003-20. IEICE, 2003.
24. Ron Steinfeld, Laurence Bull, and Yuliang Zheng. *Content Extraction Signatures*. ICISC, Volume 2288 of Lecture Notes in Computer Science, pages 285–304. Springer, 2001.