Efficient and Perfectly Unlinkable Sanitizable Signatures without Group Signatures

Christina Brzuska[1,⋆], Henrich C. Pöhls[2,4,⋆⋆], and Kai Samelin[3,4,⋆⋆⋆]

[1] Tel Aviv University, Israel
[2] Chair of IT-Security
[3] Chair of Computer Networks and Computer Communication
[4] Institute of IT-Security and Security Law (ISL), University of Passau, Germany
brzuska@post.tau.ac.il, {hp,ks}@sec.uni-passau.de

Abstract. Sanitizable signatures allow for controlled modification of signed data. The essential security requirements are accountability, privacy and unlinkability. Unlinkability is a strong notion of privacy. Namely, it makes it hard to link two sanitized messages that were derived from the same message-signature pair. In this work, we strengthen the standard unlinkability definition by *Brzuska* et al. at PKC '10, making it robust against malicious or buggy signers. While state-of-the art schemes deploy costly group signatures to achieve unlinkability, our construction uses standard digital signatures, which makes them compatible with existing infrastructure.

We construct a sanitizable signature scheme that satisfies the strong notion of perfect unlinkability and, simultaneously, achieves the strongest notion of accountability, i.e., non-interactive public accountability. Our construction is not only legally compliant, but also highly efficient, as the measurements of our reference implementation show. Finally, we revisit the security model by *Canard* et al. and correct a small flaw in their security definition given at AfricaCrypt '12.

1 Introduction

Sanitizable signature schemes (SanSigs), introduced by *Ateniese* et al. [3], enable a designated party, the sanitizer, to alter a signed document in a controlled way. The sanitizer (holding its own sanitizer secret) can generate a new, yet valid, signature for the modified document without interacting with the signer of the

⋆ Was supported by the Israel Science Foundation (grant 1076/11 and 1155/11), the Israel Ministry of Science and Technology (grant 3-9094), and the German-Israeli Foundation for Scientific Research and Development (grant 1152/2011).
⋆⋆ Is funded by BMBF (FKZ:13N10966) and ANR as part of the ReSCUeIT project.
⋆⋆⋆ The research leading to these results was supported by "Regionale Wettbewerbsfähigkeit und Beschäftigung", Bayern, 2007-2013 (EFRE) as part of the SECBIT project (http://www.secbit.de) and the European Community's Seventh Framework Programme through the EINS Network of Excellence (grant agreement no. [288021]).

S. Katsikas and I. Agudo (Eds.): EuroPKI 2013, LNCS 8341, pp. 12–30, 2014.

original document. In particular, let the message m consists of ℓ blocks, i.e., $m = (m[1], \ldots, m[\ell])$, where $\ell \in \mathbb{N}$ and $m[i] \in \{0,1\}^*$. Then, the sanitizer is only able to modify those blocks $m[i]$ that the signer defined as admissible. Sanitization thus yields a new message-signature pair (m', σ'), where σ' is a valid signature for m' under the signer's public key pk_{sig} and m' is equal to m on all *non*-admissible blocks.

Motivation. Malleable signatures of this kind seem to bear an inherent risk: a semi-trusted party is allowed to change signed data and thus a signer gives up control over the statements that are produced in its name. However, when carefully implemented, delegation of signing rights turns out to be very useful for a variety of application scenarios, ranging from sanitizing medical records to secure routing and blank signatures [3,9,22,23]. Another application scenario is access control for databases. In any larger company, but in particular in banks and hospitals, access policies are inherent in day-to-day operations [39]. Compliance rules in banks actually enforce the separation of different sectors, and likewise, hospitals host large databases of sensitive data that must not be accessed by anybody. The reception desk personnel in a hospital, for example, must not be able to access medical data of a patient, while the accountant of the hospital, in turn, must not learn personally identifying details of the patient. On the other hand, integrity of the database is crucial for both, hospitals and banks, and besides appropriate read-and-write policies [39], one might aim for the cryptographic protection of, say, patient records and have them digitally signed by the treating personnel. The accountant and the receptionist then both access different parts of an authenticity protected record. Hence, some entries in the signed record need to be sanitized while keeping a valid signature over the rest of the record. Whereas standard signatures do not allow for such modifications, sanitizable signatures enable to implement privacy-friendly access and integrity verifiability simultaneously. In particular, using sanitizable signatures, the database can operate without repeated interaction with the signer (the medical personnel). In a bank, interaction with the signer might even be disallowed, e.g., due to money laundering policies. To sum up, we aim for signatures that allow for controlled modification of different parts of a signed document without interaction with the original signer.

Sanitizable signatures usually strive for strong privacy guarantees to hide the removed sensitive information. The strongest notion of privacy is unlinkability, which we review next. In the hospital database example, we derive two different sanitized documents from the original patient's record, as we removed different parts from the same signed patient record to create a version for the accountant and another one for the reception desk personnel. A secure solution must prevent inferring information about the original record by combining the two sanitized documents, as this would violate the patient's privacy concerns as well as data-protection regulations such as HIPAA [15]. Thus, in such application domains [10], it is important for sanitizable signatures schemes to achieve unlinkability.

A digital signature on a document usually provides a legal value of evidence [34]. For example, in a hospital, the medical personnel sign their entries in the database and can be held accountable for those later, i.e., the signature allows for identifying the doctor or nurse that generated an entry. For SanSigs there are two different options for accountability: interactive accountability involves the signer and allows an authority to trace back the origin of a message-signature pair, while the scheme itself might be transparent, i.e., the origin of a signature is hidden from third-parties[1]. The alternative is a non-interactive public form of accountability [11] where third-parties can identify immediately whether the medical record was sanitized or not without interaction with the signer. As observed by *Pöhls* and *Höhne* [34], current legislation only attributes a high value of evidence to sanitizable signature if *any* subsequent change can be detected, which is incompatible with the property of transparency. Formally, *Brzuska* et al. show how to achieve public accountability in the absence of transparency, as the two are mutually exclusive [11]. In turn, privacy and a public, i.e., non-interactive, form of accountability can be achieved simultaneously, as *Brzuska* et al. [11] show. We continue this line of practical research and strengthen the privacy by achieving perfect unlinkability with standard primitives.

Challenges and Contributions. In a nutshell, our scheme addresses the following challenges: (1) Strong privacy guarantees, namely perfect unlinkability and perfect privacy, (2) a high legal value of evidence through non-interactive public accountability, (3) compatibility with existing public key infrastructures (PKI) for standard signatures, (4) performance restrictions for economic applicability. From a practitioner's point of view, having a legally recognizable *and* extremely efficient scheme is essential for deployment of a signature scheme [35,36]. Our scheme is based on the ideas given in [9,11]. It extends their work in the aforementioned points. Essentially, we achieve stronger privacy and accountability properties with simpler building blocks. Moreover, we are also able to consider unlinkability and strengthen the original definition from *Brzuska* et al. from PKC'10 [10] to be more robust against malicious or buggy signers, as well as corruption of the signer's key. Interestingly, the new definition turns out to be more compact than the original one, as the oracles that use the signer's secret key can now be simulated by the adversary. Hence, the Sign and the Proof oracle of [10] are not required anymore. We also correct a small flaw in the security definition of unlinkability in the multi-sanitizer setting given by *Canard* et al. at AfricaCrypt'12 [13]. There, the LoRSanit-oracle does not check whether *both* inputs are valid message-signature pairs and whether the requested modification is admissible. Thus, as we show, the original definition is not achievable. Finally, we show how to switch on-the-fly between unlinkability and linkability.

Techniques. The first sanitizable signature scheme [3] was based on chameleon hashes [28] that were applied per each admissible block. However, their construction allowed for mix-and-match attacks. Later schemes had to find a collision

[1] Third-parties here meaning other than signer or sanitizer.

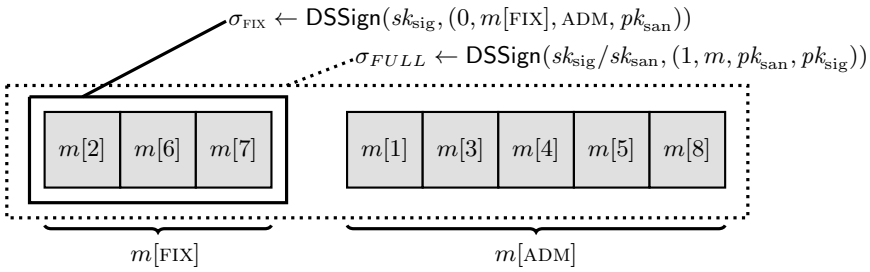

Fig. 1. Blocks 2,6,7 of m are fixed and together with pk_{san} and ADM signed by signer (σ_{FIX}). The complete message m is signed by either the signer or the sanitizer (σ_{FULL}).

on each admissible blocks, not only those modified [8]. Thereby, the sanitizing process was linear in the number of admissible blocks. Some later constructions [10,9,11] were based on a different paradigm. The idea is to use two signatures (see Fig. 1); one to sign the fix part of m, i.e., $m[\text{FIX}]$, we sometimes call this the "inner signature", and another one to sign the admissible parts, i.e., $m[\text{ADM}]$, together with the fixed parts, often called the "outer signature". The inner signature is produced by the signer of the signature scheme, while the outer signature can be produced by either one, the signer or the sanitizer. Using different signature types as inner and outer signature yields different properties of the sanitizable signature scheme. For instance, *Brzuska* et al. [10] use a group signature for the outer signature. The anonymity of the group signature makes signatures of the signer and the sanitizer indistinguishable, and the non-frameability/traceability property of the group signature scheme assures an interactive form of accountability. In turn, in [9] and [11], the authors use standard signature schemes also for the outer signature. The scheme becomes very efficient; it is not transparent anymore, but it still enjoys privacy [9,11] and a non-interactive public form of accountability [11], thus complying with legal standards. As transparency is sometimes seen as a stronger notion of privacy, one might feel that one has to compromise, as one cannot obtain a strong notion of accountability and a strong notion of privacy simultaneously. We show that this is actually not the case. As the inner signature scheme, we use a deterministic signature scheme and prove that the scheme satisfies both, a public, non-interactive version of accountability and a statistical notion of unlinkability, the strongest notion of privacy. At the same time, we maintain high efficiency and compatibility with existing public-key infrastructure, as our scheme only requires a constant number of standard building block operations. Our construction is even less complex than the ones given in [41], as we do not deploy labels. This makes our scheme applicable for use on Smart Cards [35],

embedded devices like routers, or other devices which do not have as much processing power. All of these requirements are of paramount importance to make SanSigs used in practice.

State-of-the-Art and Related Work. Malleable signatures scheme have gained a lot of attention in the past few years. They were studied in several flavors, for example, redactable signature schemes [16,25,32,33,40], sanitizable signatures with several extensions [9,12,18,22,27,30,36,42] and combinations of both approaches [24]. Moreover, the integrity protection of structured data, such as in [7,29,37,38], has equally been studied in the recent past. All schemes aim at the same goal: allowing for controlled modification of signed data to preserve privacy, while retaining authentication of origin and integrity protection against uncontrolled, i.e., unauthorized, modifications. In this paper, we focus on sanitizable signatures, as introduced by *Ateniese* et al. [3]. They also introduced the aforementioned security properties for sanitizable signatures, namely privacy, immutability, accountability, transparency and unforgeability. These were later formalized and extended by *Brzuska* et al. [8,10]. Their framework has been extended to multi-sanitizer environments by *Canard* et al. [13]. We work within the original framework for a single signer and a single sanitizer and show in Sect. 4 how to modify our scheme to the multi-user setting from [13].

The stronger notion of *statistical* unlinkability for *redactable* signature schemes was introduced by *Ahn* et al. at TCC '12 [1], and even stronger notions have been discussed recently in [4,5,19]. Neither of these notions has been considered in the context of sanitizable signatures yet, and we address this gap. The schemes for *quoting* substrings [1,4,5] are tailored towards achieving both, statistical transparency and statistical unlinkability. This ambitious goal comes at the price of weakening the unforgeability property to selective unforgeability in the case of [1]. Additionally, none of the mentioned schemes is accountable. By trading in transparency, our construction does not only achieve stronger notions of unlinkability and accountability, but also the standard adaptive unforgeability notion instead of selective unforgeability. In the context of sanitizable signatures, the notion of unlinkability captures that two sanitized *messages* cannot be linked to having the same original *message-signature pair*. For group signatures [17], in turn, the unlinkability definition corresponds to the anonymity of the signer, which is usually called transparency in the context of sanitizable signatures. The different nomenclature is maybe best explained by the fact that discussions in the area of malleable signatures are message-centered, while the way of thinking in the area of group signatures is more signer-centered—after all, the word "group" refers to a group of signers, not to a group of messages. To avoid confusion due to the historical evolution of the properties' names in the two areas, we stress that the present paper uses the nomenclature as introduced in [10]. Another related concept are proxy signatures [31]. However, they allow for delegating signing rights entirely, while sanitizable signatures allow for *altering* a specific signed message.

2 Security Models

2.1 Syntax and Notation

For a message $m = (m[1], \ldots, m[\ell])$, we call $m[i] \in \{0,1\}^*$ a *block*, while the special symbol "," $\notin \{0,1\}^*$ denotes a uniquely reversible concatenation of strings. The special symbol $\perp \notin \{0,1\}^*$ denotes an error or an exception. The following nomenclature is adapted from *Brzuska* et al. [8], who address a setting of single signers and sanitizers. We also elaborate in Sect. 4 on how to extend our construction for multi-sanitizer environments as described by *Canard* et al. [13].

Definition 1 (Sanitizable Signature Scheme). *Any* SanSig *consists of at least seven efficient, i.e., $\mathcal{PPT}$ algorithms. In particular, let* $\mathsf{SanSig} := (\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$*, 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, based on the security parameter λ:*

$$(\mathrm{pk}_{sig}, \mathrm{sk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\lambda) \quad (\mathrm{pk}_{san}, \mathrm{sk}_{san}) \leftarrow \mathsf{KGen}_{san}(1^\lambda)$$

Signing. *The* Sign *algorithm takes as input the security parameter λ, a message $m = (m[1], \ldots, m[\ell])$, $m[i] \in \{0,1\}^*$, 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 blocks. In detail,* ADM *contains a set of indices of the modifiable blocks and the overall number ℓ of blocks in m, to guard against length-altering attacks. The* Sign *algorithm outputs the message m and a signature σ (or $\perp$, indicating an error):*

$$(m, \sigma) \leftarrow \mathsf{Sign}(1^\lambda, m, \mathrm{sk}_{sig}, \mathrm{pk}_{san}, \mathrm{ADM})$$

Sanitizing. *The algorithm* Sanit *takes a message $m = (m[1], \ldots, m[\ell])$, where $m[i] \in \{0,1\}^*$, a signature σ, the security parameter λ, 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*, which contains pairs $(i, m[i]')$ describing the index i and the blocks new value $m[i]'$. We use the shorthand notation of $m' \leftarrow \mathrm{MOD}(m)$ to denote that the modification instructions were successfully applied to create the modified m' from m. Note,* MOD *could also be empty, i.e., $m' = m$ is generally possible.* Sanit *calculates a new signature σ' for the modified message $m' \leftarrow \mathrm{MOD}(m)$. The* Sanit *algorithm outputs m' and σ' (or possibly $\perp$ in case of an error).*

$$(m', \sigma') \leftarrow \mathsf{Sanit}(1^\lambda, m, \mathrm{MOD}, \sigma, \mathrm{pk}_{sig}, \mathrm{sk}_{san})$$

Verification. *The algorithm* Verify *outputs a decision $d \in \{0,1\}$ verifying the correctness of a signature σ for a message $m = (m[1], \ldots, m[\ell])$, $m[i] \in \{0,1\}^*$ with respect to the public keys pk_{sig} and pk_{san} and the security parameter λ:*

$$d \leftarrow \mathsf{Verify}(1^\lambda, m, \sigma, \mathrm{pk}_{sig}, \mathrm{pk}_{san})$$

Proof. *The algorithm* Proof *takes as input the security parameter, the secret signing key* sk_{sig}*, a message* $m = (m[1], \ldots, m[\ell])$*,* $m[i] \in \{0,1\}^*$ *and a signature* σ *as well a set of (polynomially bounded) additional message-signature pairs* $\{(m_i, \sigma_i) \mid i \in \mathbb{N}\}$ *and the public key* pk_{san}*. The* Proof *algorithm outputs a string* $\pi \in \{0,1\}^*$ *(or* $\perp$*, indicating an error):*

$$\pi \leftarrow \mathsf{Proof}(1^\lambda, \mathrm{sk}_{sig}, m, \sigma, \{(m_i, \sigma_i) \mid i \in \mathbb{N}\}, \mathrm{pk}_{san})$$

Judge. *The algorithm* Judge *takes as input the security parameter, a message* $m = (m[1], \ldots, m[\ell])$*,* $m[i] \in \{0,1\}^*$ *and a valid signature* σ*, the public keys of the parties and a proof* π*. The* Judge *algorithm outputs a decision* $d \in \{\mathtt{Sig}, \mathtt{San}, \perp\}$ *indicating whether the message-signature pair has been created by the signer or the sanitizer (or* $\perp$*, indicating an error):*

$$d \leftarrow \mathsf{Judge}(1^\lambda, m, \sigma, \mathrm{pk}_{sig}, \mathrm{pk}_{san}, \pi)$$

We require that for every SanSig the usual correctness properties hold. That is, for every genuinely created message-signature pair created by the Sign algorithm, the Verify algorithm outputs 1 with overwhelming probability. From every message-signature pair with a positive verification result, the Sanit algorithm produces again a message-signature pair for which the Verify algorithm outputs 1 with overwhelming probability. The latter is recursive, i.e., sanitized message-signature pairs can be sanitized again. For every genuinely generated value of the proof π, the Judge algorithm outputs the correct party, i.e., `Sig` or `San`, with overwhelming probability. For more details on the definition of correctness, refer to [8]. In case of a SanSig with non-interactive public accountability, the Proof algorithm always returns $\perp$, and Judge correctly decides upon such an empty proof ($\pi = \perp$) with overwhelming probability.

We write $m[\text{ADM}]$ for the uniquely reversible concatenation of the *admissible* blocks and $m[\text{FIX}]$ for the uniquely reversible concatenation of the *fixed* blocks in the order of appearance in m. Note, we do not attach any labels to the blocks as done in [41]. We require the public key to be efficiently derivable from its corresponding secret key. Additionally, we require that ADM is always correctly recoverable from any valid signature σ, which accounts for the findings by [21].

2.2 Security of Sanitizable Signatures

We present an extended security model based on [8]. It covers the basic ideas and features of a SanSig.

Definition 2 (Privacy). *A sanitizable signature scheme* SanSig *is private, if for any efficient algorithm* $\mathcal{A}$ *the probability that the experiment* $\mathsf{Privacy}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ *given in Fig. 2 returns* 1*, is negligibly close to* $\frac{1}{2}$ *(as a function of* λ*). Here, the adversary must be able to decide which input was chosen by the* LoRSanit*. The oracle signs and sanitizes the data itself.*

Experiment $\mathsf{Privacy}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$(pk_{\text{san}}, pk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$b \leftarrow \{0,1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sign}(sk_{\text{sig}},\cdots),\mathsf{Sanit}(\cdots,sk_{\text{san}})}_{\mathsf{Proof}(sk_{\text{sig}},\cdots),\mathsf{LoRSanit}(\cdots,sk_{\text{sig}},sk_{\text{san}},b)}(pk_{\text{sig}}, pk_{\text{san}})$
where oracle $\mathsf{LoRSanit}$ on input of:
$m_{0,i}, \text{MOD}_{0,i}, m_{1,i}, \text{MOD}_{1,i}, \text{ADM}_i$
if $\text{MOD}_{0,i} \not\subseteq \text{ADM}_i$, return $\perp$
if $\text{MOD}_{1,i} \not\subseteq \text{ADM}_i$, return $\perp$
if $\text{MOD}_{0,i}(m_{0,i}) \neq \text{MOD}_{1,i}(m_{1,i})$, return $\perp$
let $(m_i, \sigma_i) \leftarrow \mathsf{Sign}(m_{b,i}, sk_{\text{sig}}, pk_{\text{san}}, \text{ADM}_i)$
return $(m'_i, \sigma'_i) \leftarrow \mathsf{Sanit}(m_i, \text{MOD}_{b,i}, \sigma, pk_{\text{sig}}, sk_{\text{san}})$
return 1, if $a = b$

Fig. 2. Privacy

Experiment $\mathsf{Transparency}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$b \leftarrow \{0,1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sign}(sk_{\text{sig}},\cdots),\mathsf{Sanit}(\cdots,sk_{\text{san}})}_{\mathsf{Proof}(sk_{\text{sig}},\cdots),\mathsf{Sanit/Sign}(\cdots,sk_{\text{sig}},sk_{\text{san}},b)}(pk_{\text{sig}}, pk_{\text{san}})$
where oracle $\mathsf{Sanit/Sign}$ for input $m_i, \text{MOD}_i, \text{ADM}_i$
let $(m_i, \sigma_i) \leftarrow \mathsf{Sign}(1^\lambda, m_i, sk_{\text{sig}}, pk_{\text{san}}, \text{ADM}_i)$,
compute $(m'_i, \sigma'_i) \leftarrow \mathsf{Sanit}(m_i, \text{MOD}_i, \sigma_i, pk_{\text{sig}}, sk_{\text{san}})$
if $b = 1$:
compute $(m'_i, \sigma'_i) \leftarrow \mathsf{Sign}(m'_i, sk_{\text{sig}}, pk_{\text{san}}, \text{ADM}_i)$
finally return (m'_i, σ'_i).
return 1 if $a = b$ and $\mathcal{A}$ has not queried
any (m_i, σ_i) output by $\mathsf{Sanit/Sign}$ to Proof.

Fig. 3. Transparency

Experiment $\mathsf{Unlinkability}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$(pk_{\text{san}}, pk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$b \leftarrow \{0,1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sign}(sk_{\text{sig}},\cdots),\mathsf{Sanit}(\cdots,sk_{\text{san}})}_{\mathsf{Proof}(sk_{\text{sig}},\cdots),\mathsf{LoRSanit}(\cdots,sk_{\text{san}},sk_{\text{sig}},b)}(pk_{\text{sig}}, pk_{\text{san}})$
where oracle $\mathsf{LoRSanit}$ on input of:
$m_{0,i}, \text{MOD}_{0,i}, \sigma_{0,i}, m_{1,i}, \text{MOD}_{1,i}, \sigma_{1,i}$
//ADM needs to be recoverable from all σ
if $\text{ADM}_{0,i} \neq \text{ADM}_{1,i}$, return $\perp$
if $\text{MOD}_{0,i} \not\subseteq \text{ADM}_{0,i}$, return $\perp$
if $\text{MOD}_{1,i} \not\subseteq \text{ADM}_{1,i}$, return $\perp$
if $\text{MOD}_{0,i}(m_{0,i}) \neq \text{MOD}_{1,i}(m_{1,i})$, return $\perp$
if $\mathsf{Verify}(1^\lambda, m_{0,i}, \sigma_{0,i}, pk_{\text{sig}}, pk_{\text{san}}) \neq 1$ or
$\mathsf{Verify}(1^\lambda, m_{1,i}, \sigma_{1,i}, pk_{\text{sig}}, pk_{\text{san}}) \neq 1$, return $\perp$
return $(m', \sigma') \leftarrow \mathsf{Sanit}(m_{b,i}, \text{MOD}_{b,i}, \sigma_{b,i}, pk_{\text{sig}}, sk_{\text{san}})$
return 1, if $a = b$

Fig. 4. Unlinkability by *Brzuska* et al.

Experiment $\mathsf{UnlinkabilityEx}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda, n, m)$
$(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Setup}(1^\lambda, n, m)$
$b \leftarrow \{0,1\}$
$(m_0, \text{MOD}_0, \sigma_0, m_1, \text{MOD}_1, \sigma_1, j_0, j_1, st) \leftarrow \mathcal{A}^{(*)}_{ch}(\mathsf{PK})$
let $(m', \sigma') \leftarrow \mathsf{Sanit}(m_b, \sigma_b, \mathsf{SK}[j_b], \text{MOD}_b, \mathsf{PK})$
let $(\mathsf{ORI}, I_{\text{ORI}}, \pi_{ORI}) \leftarrow \mathsf{FindOri}(m', \sigma'_b, \mathsf{osk}_{\text{ORI}}, \mathsf{PK})$
if $I_{\text{ORI}} = 0$ or $(I_{\text{ORI}} = (\mathsf{Sig}, i_{\text{ORI}})$ and $i_{\text{ORI}} \in \mathcal{CU})$ or
$j_0 \in \mathcal{CU}$ or $j_1 \in \mathcal{CU}$ or
$\mathsf{Judge}(m', \sigma'_b, (\mathsf{ORI}, i_{\text{ORI}}, \pi_{\text{ORI}}), \mathsf{PK}) = 0$, return $\perp$
let $b^* \leftarrow \mathcal{A}^{(*)}_{gu}(m', \sigma'_b, st)$
if (m', σ') was queried to $\mathsf{SigOpen}$, return $\perp$, else b^*

Fig. 5. Unlinkability by *Canard* et al.

Experiment $\mathsf{SUnlinkability}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda)$
$(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$b \leftarrow \{0,1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sanit}(\cdots,sk_{\text{san}}),\mathsf{LoRSanit}(\cdots,sk_{\text{san}},b)}(pk_{\text{san}})$
where oracle $\mathsf{LoRSanit}$ on input of:
$m_{0,i}, \text{MOD}_{0,i}, \sigma_{0,i}, m_{1,i}, \text{MOD}_{1,i}, \sigma_{1,i}, pk_{\text{sig},i}$
//ADM needs to be recoverable from all σ
if $\text{ADM}_{0,i} \neq \text{ADM}_{1,i}$, return $\perp$
if $\text{MOD}_{0,i} \not\subseteq \text{ADM}_{0,i}$, or $\text{MOD}_{1,i} \not\subseteq \text{ADM}_{1,i}$, or $\text{MOD}_{0,i}(m_{0,i}) \neq \text{MOD}_{1,i}(m_{1,i})$, return $\perp$
if $\mathsf{Verify}(1^\lambda, m_{0,i}, \sigma_{0,i}, pk_{\text{sig},i}, pk_{\text{san}}) \neq 1$ or $\mathsf{Verify}(1^\lambda, m_{1,i}, \sigma_{1,i}, pk_{\text{sig},i}, pk_{\text{san}}) \neq 1$, return $\perp$
return $(m', \sigma') \leftarrow \mathsf{Sanit}(m_{b,i}, \text{MOD}_{b,i}, \sigma_{b,i}, pk_{\text{sig},i}, sk_{\text{san}})$
return 1, if $a = b$

Fig. 6. Strengthened Unlinkability

Definition 3 (Transparency). *A sanitizable signature scheme SanSig is transparent, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Transparency}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda)$ given in Fig. 3 returns 1, is negligibly close to $\frac{1}{2}$ (as a function of λ). Here, the oracle either directly signs the expected output message ($b = 1$) or signs the input and then sanitizes it to the expected output ($b = 0$).*

Please note, in accordance with *Brzuska* et al. [8], we require that $\text{MOD}_i \subseteq \text{ADM}_i$ is true. We check this for all queries to the above oracles as otherwise a trivial attack vector exists. In particular, if $\text{MOD}_i \not\subseteq \text{ADM}_i$ yields that Sanit outputs $\perp$ then this is easily distinguishable from a "fresh" signature.

Definition 4 (Strengthened Unlinkability following *Brzuska* et al.). *A sanitizable signature scheme SanSig is* unlinkable*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{SUnlinkability}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ given in Fig. 6 returns* 1 *is negligibly close to $\frac{1}{2}$ (as a function of λ). Here, the adversary has to guess which of the two inputted message-signature pairs was chosen to be sanitized.*

Compare our new definition of unlinkability in Fig. 6 with the original definition of [10] depicted in Fig. 4. We altered the LoRSanit oracle so that it does not specify the signer by having a fixed signer key pk_{sig}. In turn, now, the adversary chooses pk_{sig}. Intuitively, this captures that unlinkability holds as long as the sanitizer is honest, even if the signer happens to be dishonest. One might argue that a malicious signer can always break any privacy or unlinkability property, as it knows which messages it signed and thus, it can always recover the originally signed message. However, there exist intermediary stages, for example, if the signer is buggy, uses weak randomness, or loses its secret key. In these cases, our definition turns out to be robust, i.e., even if the signer loses its secret key, unlinkability and therefore privacy are preserved. In a sense, we cover the equivalent of forward secrecy for the case of key exchange: there, previous sessions remain secure when long-term secrets are lost [14].

Actually, we even achieve a stronger notion of secrecy than key exchange can attain. Unlinkability is even preserved when the signer loses his secret key *before* signing the message and even when using some form of "bad" secret key specified by the adversary. In Def. 4, there are no signing and proof oracles—the reason is that the adversary can now simulate those by itself. We now prove that our new notion of unlinkability is strictly stronger than the original one by *Brzuska* et al. [10].

Theorem 1. *Any strongly unlinkable SanSig is also unlinkable. The converse is not true.*

Proof. Let SanSig be a strongly unlinkable sanitizable signature scheme. We prove via reduction that SanSig is also unlinkable. Let $\mathcal{A}$ be an adversary against the unlinkability of SanSig. Using $\mathcal{A}$, we can construct an adversary $\mathcal{B}$ against the strong unlinkability of SanSig as follows. In the beginning of the game, $\mathcal{B}$ relays the sanitizer's public key to $\mathcal{A}$ and runs the key generation algorithm of the signer and returns the signer's public key pk_{sig} to $\mathcal{A}$. The signer's secret key is used to answer all queries that $\mathcal{A}$ makes to Sign and Proof. The remaining queries are queries to the Sanit oracle, which $\mathcal{B}$ simply relays, and queries to the LoRSanit oracle which $\mathcal{B}$ prepends with the signer's public key pk_{sig} and queries to its own LoRSanit oracle. In the end, $\mathcal{A}$ returns a bit that $\mathcal{B}$ returns as well. As the simulation is perfect, $\mathcal{B}$'s advantage against strong unlinkability is as big as $\mathcal{A}$'s advantage against unlinkability.

We now separate the two unlinkability notions by constructing a (contrived) scheme that fulfills the security requirements of the original security model by *Brzuska* et al. [10] but is insecure in our new model.

Let $\mathsf{SanSig} = (\mathsf{KGen}_{\mathrm{sig}}, \mathsf{KGen}_{\mathrm{san}}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$ be a secure sanitizable signature scheme. To obtain a counterexample, we adjust the scheme as follows:

- $\mathsf{KGen}'_{\mathrm{sig}}$ works as $\mathsf{KGen}_{\mathrm{sig}}$, except that it appends a 1 to the public key returned by $\mathsf{KGen}_{\mathrm{sig}}$
- $\mathsf{KGen}'_{\mathrm{san}}$ works as its original counterpart
- Judge', Sign', and Verify' work as their original counterparts, but cut of the last bit of pk_{sig}
- Sanit' is the same as Sanit with one exception: if pk_{sig} ends with a 0, it proceeds as normal, while cutting of the last bit of pk_{sig}. Otherwise, it outputs the *original* signature and message, instead of sanitizing it.

Our stronger attacker, that can choose pk_{sig} by running $\mathsf{KGen}'_{\mathrm{sig}}$, now wins by replacing the trailing 1 with a 0. However, the scheme is fully secure in the original definition where an attacker was not able to influence pk_{sig}. Here, it does not matter, if the keys are indistinguishable. All other properties are not affected and are inherited from the original SanSig. Note, this scheme is also still correct as the message-signature pair left untouched by the modified Sanit' still verifies, while also $\mathrm{MOD}_1(m_1) = \mathrm{MOD}_2(m_2)$ yields.

As our stronger notion of unlinkability implies the original notion of unlinkability, all known implications still hold. In particular, our strengthened unlinkability also implies privacy [10].

Definition 5 (Unlinkability by *Canard* et al.). *A sanitizable signature scheme SanSig is* unlinkable*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{UnlinkabilityEx}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ given in Fig. 6 returns* 1 *is negligibly close to $\frac{1}{2}$ (as a function of λ). The basic idea is that an adversary has to guess which message-signatures pair from the two inputs was chosen to be sanitized.*

For self-containment, we introduce the notation of the complete multi-sanitizer framework and refer to *Canard et al.* [13] for a complete discussion. We have a choose-then-guess adversary, which state is denoted as *st*. The algorithms and oracles are defined as follows:
(1) $\mathsf{Setup}(1^\lambda, n, m)$ is the instance generator, where n denotes the amount of signers, while m denotes the number of sanitizers. Hence, PK contains all public keys, while SK contains all private keys. The adversary $\mathcal{A}$ also gains access to sanitization, sign, proof and all "opening" oracles:
(2) $\mathsf{FindOri}$ returns the index of the original *signer*. If the message has been sanitized, $\mathsf{FindOri}$ returns 0, as "no signer exists". It requires the opening key $\mathsf{osk}_{\mathrm{ORI}}$.

Experiment $\mathsf{Pubaccountability}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$(pk^*, m^*, \sigma^*, \pi^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\text{sig}}, \cdot, \cdot)}_{\mathsf{Sanit}(\ldots, sk_{\text{san}})}(pk_{\text{sig}}, pk_{\text{san}})$
Let $(m_i, \text{ADM}_i, pk_{\text{san},i})$ and σ_i for $i = 1, 2, \ldots, q$ be the queries to and from oracle Sign
return 1 if
$\forall i : (pk^*, m^*) \neq (pk_{\text{san},i}, m_i)$, and
$\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk_{\text{sig}}, pk^*) = 1$, and
$\mathsf{Judge}(1^\lambda, m^*, \sigma^*, pk_{\text{sig}}, pk^*, \bot) = \mathtt{Sig}$
Let $(m_j, \text{MOD}_j, \sigma_j, pk_{\text{sig},j})$ and (m'_j, σ'_j) be the queries to/from oracle Sanit
return 1 if:
$\forall j : (pk^*, m^*) \neq (pk_{\text{sig},j}, m'_j)$, and
$\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\text{san}}) = 1$, and
$\mathsf{Judge}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\text{san}}, \bot) = \mathtt{San}$
return 0

Fig. 7. Non-Interactive Public Accountability

Experiment $\mathsf{Immutability}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$(pk^*_{\text{san}}, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(sk_{\text{sig}}, \cdots), \mathsf{Proof}(sk_{\text{sig}}, \cdots)}(pk_{\text{sig}})$
let $(m_i, \text{ADM}_i, pk_{\text{san},i})$ and σ_i, $i = 1, 2, \ldots, q$
denote the queries to Sign
return 1, if:
$\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk_{\text{sig}}, pk^*_{\text{san}}) = 1$, and
for all $i = 1, 2, \ldots, q$ we have:
$pk^*_{\text{san}} \neq pk_{\text{san},i}$, or
$m^*[j_i] \neq m_i[j_i]$, where $j_i \notin \text{ADM}_i$
shorter messages are padded with $\bot$

Fig. 8. Immutability

(3) $\mathcal{CU}$ denotes the set of corrupted participants, i.e., all signers and sanitizers which secret key is known to the adversary. The adversary can gain access to the secret keys by using an implicit Corrupt oracle.
(4) Judge works as in our original definition, while it accounts for multiple signer, and sanitzers resp., and allows "partial" openings. In particular, it gets an additional parameter ORI. It outputs the index of the original signer, if the message has *not* been sanitized. See [13] for a complete discussion.

Our main observation is that it is crucial for the left-or-right oracle to check the validity of both signatures before proceeding. Else, the definition is not satisfiable and would not be satisfied by the scheme in [13]. The problem is that the left-or-right oracle receives two message-signature pairs and will sanitize one of them. If one of the signature is empty, then the sanitizing algorithm is unable to produce a valid signature because else, the sanitizer would be able to break immutability. Thus, if the adversary gives a valid message-signature pair to the oracle and a pair of a message and an empty (and thus invalid) signature, then in one case, the oracle returns $\bot$, and in the other case, the oracle returns a valid answer. Thus, the adversary can distinguish the two cases.

Definition 6 (Non-Interactive Public Accountability). *A sanitizable signature scheme SanSig is* non-interactive public accountable*, if for an empty proof $\pi = \bot$, and for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Pubaccountability}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ given in Fig. 7 returns 1 is negligible (as a function of λ). The basic idea is that an adversary, i.e., the sanitizer or the signer, has to be able to make the Judge decide wrongly on an empty proof $\pi = \bot$. Note, Proof always returns $\bot$ and therefore is not an oracle here.*

Definition 7 (Immutability). *A sanitizable signature scheme SanSig is immutable, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Immutability}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ given in Fig. 8 returns 1, is negligible (as a function of λ). The basic idea is, that an adversary is not able to modify non-admissible blocks, even if it is able to choose the sanitizer key pair.*

Definition 8 (Secure SanSig). *We call a SanSig secure, if it is unlinkable, immutable and non-interactive publicly accountable.*

Note, unlinkability implies privacy, while non-interactive public accountability implies accountability [11] and therefore also unforgeability [8]. Recall the following separation by *Brzuska* et al. [10], which also applies for our strengthened unlinkability definition, as the latter implies the original definition of unlinkability in [10], as we have already proven.

Theorem 2 (Unlinkability $\not\Rightarrow$ Transparency). *There exists a scheme which is unlinkable, but not transparent.*

3 Efficient Perfectly Unlinkable SanSig

We introduce the building blocks used in the construction and then give a formal algorithmic description of our construction.

3.1 Building Blocks

This section introduces the required building blocks for our construction. We require a deterministic signature scheme, unforgeable under chosen message attacks (UNF-CMA). Let $\mathcal{DS} = (\mathsf{DSKGen}, \mathsf{DSSign}, \mathsf{DSVerify})$ be such a signature scheme. Deterministic means, that signing identical messages leads to identical signatures, if signed with the same secret key *sk*. We want to emphasize that every unforgeable signature scheme can be transformed into a strongly unforgeable and also deterministic scheme using several transformations [6,20]. An example for a standardized deterministic signature scheme is "RSASSA-PKCS-v1_5-SIGN" [26].

3.2 Algorithmic Description

Our scheme is inspired by the constructions given in [9] and [11]. It achieves unforgeability, immutability, non-interactive public accountability, perfect privacy, perfect unlinkability and sanitizer- and signer-accountability. Therefore, it meets all legal and the essential cryptographic requirements.

Construction 1 (Secure SanSig). *Let $\mathcal{DS} = (\mathsf{DSKGen}, \mathsf{DSSign}, \mathsf{DSVerify})$ be a deterministic and unforgeable signature scheme. Define the sanitizable signature scheme $\mathsf{SanSig} = (\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Judge})$ as follows:*

Key Generation: *Algorithm KGen_{sig} generates on input of the security parameter λ a key pair $(\mathrm{pk}_{sig}, \mathrm{sk}_{sig}) \leftarrow \mathsf{DSKGen}(1^\lambda)$ of the underlying signature scheme $\mathcal{DS}$, and algorithm KGen_{san} for input λ analogously returns a pair $(\mathrm{pk}_{san}, \mathrm{sk}_{san}) \leftarrow \mathsf{DSKGen}(1^\lambda)$.*

***Signing**: Algorithm* Sign *on input* $m \in \{0,1\}^*$, sk_{sig}, pk_{san}, ADM *and computes*

$$\sigma_{\mathrm{FIX}} \leftarrow \mathsf{DSSign}(\mathrm{sk}_{sig}, (0, m[\mathrm{FIX}], \mathrm{ADM}, \mathrm{pk}_{san})),$$

$$\sigma_{\mathrm{FULL}} \leftarrow \mathsf{DSSign}(\mathrm{sk}_{sig}, (1, m, \mathrm{pk}_{san}, \mathrm{pk}_{sig}))$$

using the underlying signing algorithm. It returns:

$$(m, \sigma) = (m, (\sigma_{\mathrm{FIX}}, \sigma_{\mathrm{FULL}}, \mathrm{ADM}))$$

***Sanitizing**: Algorithm* Sanit *on input of message* m*, (maybe empty) modification instructions* MOD*, a signature* $\sigma = (\sigma_{\mathrm{FIX}}, \sigma_{\mathrm{FULL}}, \mathrm{ADM})$*, keys* pk_{sig} *and* sk_{san}*, first checks that* MOD *is admissible according to* ADM *and that* σ_{FIX} *is a valid signature for message* $(0, m[\mathrm{FIX}], \mathrm{ADM}, \mathrm{pk}_{san})$ *under key* pk_{sig}*. If not, it stops and outputs* $\perp$*. Else, it generates the modified message* $m' \leftarrow \mathrm{MOD}(m)$ *and computes*

$$\sigma'_{\mathrm{FULL}} \leftarrow \mathsf{DSSign}(\mathrm{sk}_{san}, (1, m', \mathrm{pk}_{san}, \mathrm{pk}_{sig}))$$

and outputs $(m', \sigma') = (m', (\sigma_{\mathrm{FIX}}, \sigma'_{\mathrm{FULL}}, \mathrm{ADM}))$*.*

***Verification**: Algorithm* Verify *on input of a message* $m \in \{0,1\}^*$*, a signature* $\sigma = (\sigma_{\mathrm{FIX}}, \sigma_{\mathrm{FULL}}, \mathrm{ADM})$ *and public keys* pk_{sig}, pk_{san} *first checks that* σ_{FIX} *is a valid signature for message* $(0, m[\mathrm{FIX}], \mathrm{ADM}, \mathrm{pk}_{san})$ *under key* pk_{sig} *by checking that* $\mathsf{DSVerify}(\mathrm{pk}_{sig}, (0, m[\mathrm{FIX}], \mathrm{ADM}, \mathrm{pk}_{san}), \sigma_{\mathrm{FIX}}) = 1$*. Second, it returns* 1*, if:* $\mathsf{DSVerify}(\mathrm{pk}_{sig}, (1, m, \mathrm{pk}_{san}, \mathrm{pk}_{sig}), \sigma_{\mathrm{FULL}}) = 1$ *or* $\mathsf{DSVerify}(\mathrm{pk}_{san}, (1, m, \mathrm{pk}_{san}, \mathrm{pk}_{sig}), \sigma_{\mathrm{FULL}}) = 1$*. This declares the entire signature as valid. Otherwise it returns* 0*.*

***Proof:** The* Proof *algorithm always returns* $\perp$

***Judge:** Judge on input of* $m, \sigma, \mathrm{pk}_{sig}, \mathrm{pk}_{san}$ *and* $\perp$ *parses* σ *as* $(\sigma_{\mathrm{FIX}}, \sigma_{\mathrm{FULL}}, \mathrm{ADM})$ *and outputs* `Sig`*, if:*

$$\mathsf{DSVerify}(\mathrm{pk}_{sig}, (1, m, \mathrm{pk}_{san}, \mathrm{pk}_{sig}), \sigma_{\mathrm{FULL}}) = 1$$

It returns `San`*, if:*

$$\mathsf{DSVerify}(\mathrm{pk}_{san}, (1, m, \mathrm{pk}_{san}, \mathrm{pk}_{sig}), \sigma_{\mathrm{FULL}}) = 1$$

If none verifies, it returns $\perp$*.*

Theorem 3 (Our construction is secure.). *If the underlying signature scheme* DS *is UNF-CMA and deterministic, then our construction is immutable, perfectly unlinkable, perfectly private, non-interactive publicly accountable (and therefore signer-/sanitizer accountable and also unforgeable [8,11]), i.e., secure.*

The proof is delegated to Appendix A.

4 Extensions: Multiple Sanitizers and Selective Linkability

We deploy a deterministic, strongly unforgeable signature scheme to obtain unlinkability. In the following, we show that by replacing the signature scheme for the fixed blocks with certain other types of signature schemes, one obtains interesting additional features, which have not been considered yet. In this section, we extend our scheme to cope with multiple sanitizers and allow for more fine-grained control.

Multiple Sanitizer and Speed-Up. Our construction can be modified to work in the multi-sanitizer framework [13]. In particular, the signer can add a public key $pk_{\text{san},i}$ for each sanitizer i. Our simple yet effective alteration once again demonstrates the generality of the underlying basic idea and its broad applicability. Additionally, all mentioned modifications impact the performance of the scheme only lightly, as they require only a constant number of additional steps per sanitizer. To improve the speed of the multi-sanitizer verification procedure, one can append a hint on the required public key to the signature. This small improvement allows skipping the need to iterate through all public keys in σ_{FIX}. This also holds when considering only one sanitizer and one signer. For a meaningful definition of unlinkability in the multi-sanitizer case, we require that only one sanitizer acts as the sanitizer in the unlinkability experiment, while we also require that the sanitizers are fixed. The latter is in conjunction with [13]. To enhance anonymity for the sanitizers, one can also use a group signature scheme for the *sanitizers*. As the original signer still uses a normal signature scheme, the signatures are clearly distinguishable and therefore remain legally recognized, following the reasoning of [34]. In certain scenarios, the weakened unlinkability definition still suffices, if it is obvious in the practical application which entity has generated the signature.

Signer Selected Linkability by Strongly Unforgeable Signatures. A strongly unforgeable signature scheme is an unforgeable signature scheme, where, additionally, it is hard to generate new signatures for previously signed messages [2]. If the signature generation of a strongly unforgeable scheme is randomized, then signatures for a message are not unique which harms unlinkability in our construction. Thus, if we de-randomize the linkable scheme by using a $\mathcal{PRF}$ [20] to generate the randomness for the signing algorithm deterministically from the input message, then the signer is able to make that signature unlinkable. In detail, we consider the $\mathcal{PRF}$-key as part of sk_{sig}. Then, if two signatures are designated to be unlinkable, the used randomness is deterministically generated by applying the $\mathcal{PRF}$ on a digest of the fixed part of the message that is generated using a collision-resistant hash function. On the other hand, if the used randomness is not generated using the $\mathcal{PRF}$, the resulting sanitized documents can be linked. Overall, the $\mathcal{PRF}$-de-randomized scheme allows the signer to decide whether signed messages shall be linkable or unlinkable. It remains to establish formally that these properties hold.

Sanitizer Selective Linkability by Randomizable Signatures. As already proposed by *Brzuska* et al., re-randomizable signatures are also suitable to achieve unlinkability [10]. Interestingly, here, a dual observation to the $\mathcal{PRF}$-case applies. Namely, the sanitizer gets to choose whether a sanitized message shall be linkable to the original one or not. Note, this feature comes with the caveat that the signer relies on good randomness added by the sanitizer. If the sanitizing process is carried out by a weak or only partially trusted device, one might prefer to opt against re-randomizable signatures and use deterministic

Table 1. Median runtime of our scheme; ℓ is the number of blocks; All in ms

	KeyGen	Sign			Sanit of 25% of ℓ			Verify			Detect		
λ \ ℓ	100/1k/10k	100	1k	10k	100	1k	10k	100	1k	10k	100	1k	10k
2.048 Bit	1,934	22	24	26	13	14	17	1	1	4	1	2	5
4.096 Bit	16,280	149	150	149	78	79	84	4	4	8	5	5	9

ones, as proposed in our construction. On the other hand, it allows the signer to delegate the decision to enable linkability for certain messages to the sanitizer.

5 Performance Measurements

To demonstrate practical usability, we implemented our construction. All tests were performed on an *Intel* T8300 Dual Core @`2.40` GHz and `4` GiB of RAM, running *Ubuntu* Version 12.04 LTS (64 Bit) and Java version `1.7.0_03`. For all tests, we applied our algorithms to messages with 100, 1,000 (1k) and 10,000 (10k) blocks. For any block count, we decided to fix the amount of admissible blocks to 50%, and we sanitized always 50% of the admissible blocks, i.e., 25% of all blocks. We took the median of `100` runs. We used one possible choice for a deterministic signature scheme, namely "RSASSA-PKCS-v1_5-SIGN" [26]. We utilized a single thread to calculate the signatures. Obviously, parallelization, e.g., by using CRT, will yield significant performance improvements. The results of our measurements in Tab. 1 show that our scheme keeps a very high performance. The source code is available upon request.

References

1. Ahn, J.H., Boneh, D., Camenisch, J., Hohenberger, S., Shelat, A., Waters, B.: Computing on authenticated data. ePrint Report 2011/096 (2011)
2. An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 83–107. Springer, Heidelberg (2002)
3. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: de Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
4. Attrapadung, N., Libert, B., Peters, T.: Computing on authenticated data: New privacy definitions and constructions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 367–385. Springer, Heidelberg (2012)
5. Attrapadung, N., Libert, B., Peters, T.: Efficient completely context-hiding quotable and linearly homomorphic signatures. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 386–404. Springer, Heidelberg (2013)
6. Bellare, M., Shoup, S.: Two-tier signatures, strongly unforgeable signatures, and fiat-shamir without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 201–216. Springer, Heidelberg (2007)

7. Brzuska, C., et al.: Redactable Signatures for Tree-Structured Data: Definitions and Constructions. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 87–104. Springer, Heidelberg (2010)
8. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of Sanitizable Signatures Revisited. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 317–336. Springer, Heidelberg (2009)
9. Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Sanitizable signatures: How to partially delegate control for authenticated data. In: Proc. of BIOSIG. LNI, vol. 155, pp. 117–128. GI (2009)
10. Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Unlinkability of Sanitizable Signatures. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 444–461. Springer, Heidelberg (2010)
11. Brzuska, C., Pöhls, H.C., Samelin, K.: Non-Interactive Public Accountability for Sanitizable Signatures. In: De Capitani di Vimercati, S., Mitchell, C. (eds.) EuroPKI 2012. LNCS, vol. 7868, pp. 178–193. Springer, Heidelberg (2013)
12. Canard, S., Jambert, A.: On extended sanitizable signature schemes. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 179–194. Springer, Heidelberg (2010)
13. Canard, S., Jambert, A., Lescuyer, R.: Sanitizable signatures with several signers and sanitizers. In: Mitrokotsa, A., Vaudenay, S. (eds.) AFRICACRYPT 2012. LNCS, vol. 7374, pp. 35–52. Springer, Heidelberg (2012)
14. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
15. Caplan, R.M.: HIPAA. Health Insurance Portability and Accountability Act of 1996. Dent Assist 72(2), 6–8 (1997)
16. Chang, E.-C., Lim, C.L., Xu, J.: Short Redactable Signatures Using Random Trees. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 133–147. Springer, Heidelberg (2009)
17. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
18. de Meer, H., Pöhls, H.C., Posegga, J., Samelin, K.: Scope of security properties of sanitizable signatures revisited. In: ARES 2013, pp. 188–197 (2013)
19. Deiseroth, B., Fehr, V., Fischlin, M., Maasz, M., Reimers, N.F., Stein, R.: Computing on authenticated data for adjustable predicates. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 53–68. Springer, Heidelberg (2013)
20. Goldreich, O.: Two remarks concerning the goldwasser-micali-rivest signature scheme. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 104–110. Springer, Heidelberg (1987)
21. Gong, J., Qian, H., Zhou, Y.: Fully-secure and practical sanitizable signatures. In: Lai, X., Yung, M., Lin, D. (eds.) Inscrypt 2010. LNCS, vol. 6584, pp. 300–317. Springer, Heidelberg (2011)
22. Haber, S., Hatano, Y., Honda, Y., Horne, W.G., Miyazaki, K., Sander, T., Tezoku, S., Yao, D.: Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In: ASIACCS, pp. 353–362 (2008)
23. Hanser, C., Slamanig, D.: Blank digital signatures. In: AsiaCCS, pp. 95–106. ACM (2013)
24. Izu, T., Kunihiro, N., Ohta, K., Sano, M., Takenaka, M.: Sanitizable and Deletable Signature. In: Chung, K.-I., Sohn, K., Yung, M. (eds.) WISA 2008. LNCS, vol. 5379, pp. 130–144. Springer, Heidelberg (2009)

25. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
26. Jonsson, J., Kaliski, B.: Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447 (Informational) (February 2003)
27. Klonowski, M., Lauks, A.: Extended Sanitizable Signatures. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 343–355. Springer, Heidelberg (2006)
28. Krawczyk, H., Rabin, T.: Chameleon Hashing and Signatures. In: Symposium on Network and Distributed Systems Security, pp. 143–154 (2000)
29. Kundu, A., Bertino, E.: How to authenticate graphs without leaking. In: EDBT, pp. 609–620 (2010)
30. Lai, J., Ding, X., Wu, Y.: Accountable trapdoor sanitizable signatures. In: Deng, R.H., Feng, T. (eds.) ISPEC 2013. LNCS, vol. 7863, pp. 117–131. Springer, Heidelberg (2013)
31. Mambo, M., Usuda, E., Okamoto, K.: and Okamoto. Proxy signatures for delegating signing operation. In: CCS 1996, pp. 48–57. ACM, New York (1996)
32. Miyazaki, K., Hanaoka, G., Imai, H.: Digitally signed document sanitizing scheme based on bilinear maps. In: ASIACCS 2006, pp. 343–354. ACM (2006)
33. 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), 239–246 (2005)
34. Pöhls, H.C., Höhne, F.: The role of data integrity in EU digital signature legislation — achieving statutory trust for sanitizable signature schemes. In: Meadows, C., Fernandez-Gago, C. (eds.) STM 2011. LNCS, vol. 7170, pp. 175–192. Springer, Heidelberg (2012)
35. Pöhls, H.C., Peters, S., Samelin, K., Posegga, J., de Meer, H.: Malleable signatures for resource constrained platforms. In: Cavallaro, L., Gollmann, D. (eds.) WISTP 2013. LNCS, vol. 7886, pp. 18–33. Springer, Heidelberg (2013)
36. Pöhls, H.C., Samelin, K., Posegga, J.: Sanitizable Signatures in XML Signature — Performance, Mixing Properties, and Revisiting the Property of Transparency. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 166–182. Springer, Heidelberg (2011)
37. Samelin, K., Pöhls, H.C., Bilzhause, A., Posegga, J., de Meer, H.: On Structural Signatures for Tree Data Structures. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 171–187. Springer, Heidelberg (2012)
38. Samelin, K., Pöhls, H.C., Bilzhause, A., Posegga, J., de Meer, H.: Redactable signatures for independent removal of structure and content. In: Ryan, M.D., Smyth, B., Wang, G. (eds.) ISPEC 2012. LNCS, vol. 7232, pp. 17–33. Springer, Heidelberg (2012)
39. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer 29(2), 38–47 (1996)
40. Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: Kim, K. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer, Heidelberg (2002)
41. Yum, D.H., Lee, P.J.: Sanitizable signatures reconsidered. IEICE Transactions 94-A(2), 717–724 (2011)
42. Yum, D.H., Seo, J.W., Lee, P.J.: Trapdoor sanitizable signatures made easy. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 53–68. Springer, Heidelberg (2010)

A Security of Our Construction

The security proofs of our construction follow the ideas of [9] and [11]. From [8,9,11] we yield that non-interactive public accountability, as defined in [11], already implies sanitizer accountability, signer accountability and unforgeability. Moreover, unlinkability implies privacy [9]. Thus, it suffices to prove that Construction 1 is immutable, non-interactive publicly accountable and unlinkable.

Proof (of Th. 3). To increase the readability, we prove each property on its own.

- *Unlinkability.* For two messages m^0 and m^1 with identical fixed parts $m[\text{FIX}]$, the signatures σ^0_{FIX} and σ^1_{FIX} over this part are identical, as we use a deterministic signature scheme. Moreover, the signatures σ^0_{FULL} and σ^1_{FULL}, depending on the modifiable message parts, are not used as input for the sanitizing process. Thus, we are perfectly unlinkable and perfectly private.
- *Immutability.* Assume towards contradiction that Construction 1 is not immutable. In particular, let $\mathcal{A}$ be an efficient adversary against immutability. We construct an adversary $\mathcal{B}$ against the underlying signature scheme. The adversary $\mathcal{B}$ embeds the keys of the signature scheme as the signer's public keys. It then answers $\mathcal{A}$'s queries to the signing oracle by running the algorithm as described in Construction 1, except for signature generation under the signer's key, where $\mathcal{B}$ queries its signing oracle instead of computing them itself. The simulation is perfect. When $\mathcal{A}$ returns $(m^*, pk^*_{\text{san}}, \sigma^*)$, then $\mathcal{B}$ returns $((0, m[\text{FIX}], \text{ADM}, pk_{\text{san}}), \sigma_{\text{FIX}})$ as a forgery. We now prove that $\mathcal{B}$ is successful in attacking the underlying signature scheme, if $\mathcal{A}$ is.

  Following our definition, $\mathcal{A}$ wins, if it can output a tuple $(m^*, \sigma^*, pk^*_{\text{san}})$ such that $\mathsf{Verify}(m^*, \sigma^*, pk_{\text{sig}}, pk^*_{\text{san}}) = 1$ and $pk^*_{\text{san}} \neq pk_{\text{san},i}$ for all i queries to the signing oracle or $\exists i, j, j_i \notin \text{ADM} : m^*[j_i] \neq m_i[j_i]$.

  (i) If $pk^*_{\text{san}} \neq pk_{\text{san}\,i}$, then $(0, *, *, pk^*_{\text{san}})$ is fresh.
  (ii) If $\exists i, j, j_i \notin \text{ADM} : m^*[j_i] \neq m_i[j_i]$, then $(0, m[\text{FIX}]^*, \text{ADM}, pk^*_{\text{san}})$ is fresh.

  These cases are equal to the attack cases for forgeries of the underlying signature scheme. Thus, $\mathcal{B}$'s success probability is equal to $\mathcal{A}$'s success probability.
- *Non-Interactive Public Accountability.* Let $\mathcal{A}$ be an efficient adversary against non-interactive public accountability. We construct another efficient adversary $\mathcal{B}$ against the unforgeability of the underlying signature scheme $\mathcal{DS}$ as follows. $\mathcal{B}$ gets as input a public key pk and flips a coin b. If $b = 0$, it sets $pk_{\text{sig}} := pk$ and runs DSKGen to generate $(pk_{\text{san}}, sk_{\text{san}})$. If $b = 1$, it sets $pk_{\text{san}} := pk$ and runs DSKGen to generate $(pk_{\text{sig}}, sk_{\text{sig}})$. To simulate the oracles for $\mathcal{A}$, the algorithm $\mathcal{B}$ runs the algorithms Sign and Sanit according to the specification with the exception that whenever a signature is generated under the secret key sk corresponding to pk, $\mathcal{B}$ does not generate the signature itself. Instead, $\mathcal{B}$ queries its signing oracle and passes the result to $\mathcal{A}$. Eventually, the adversary $\mathcal{A}$ outputs a triple (pk^*, m^*, σ^*). We distinguish between two cases, a malicious sanitizer attack and a malicious signer attack.

With probability $\frac{1}{2}$ the simulation was done for the correct case, as in both cases, the output distributions of $\mathcal{B}$'s simulation are identical.

Malicious Sanitizer
$\mathcal{B}$ returns $((0, m[\text{FIX}]^*, \text{ADM}, pk^*_{\text{san}}), \sigma_{\text{FIX}})$. As $m[\text{FIX}]^*$ is fresh, the signing oracle has never signed a message of the form $(0, m[\text{FIX}], \text{ADM}, *)$.

Malicious Signer
$\mathcal{B}$ returns $((1, m^*, pk_{\text{san}}, pk^*_{\text{sig}}), \sigma_{\text{FULL}})$. As m^* is fresh, the signing oracle has never signed a message of the form $(1, m^*, *, *)$.

Analysis
Thus, the overall success probability of $\mathcal{B}$ is exactly $\frac{1}{2}$ the success probability of $\mathcal{A}$. □