Invisible Sanitizable Signatures and Public-Key Encryption are Equivalent

Marc Fischlin and Patrick Harasser(✉)

Cryptoplexity, Technische Universität Darmstadt, Darmstadt, Germany
{marc.fischlin,patrick.harasser}@cryptoplexity.de
http://www.cryptoplexity.de

Abstract. Sanitizable signature schemes are signature schemes which support the delegation of modification rights. The signer can allow a sanitizer to perform a set of admissible operations on the original message and then to update the signature, in such a way that basic security properties like unforgeability or accountability are preserved. Recently, Camenisch *et al.* (PKC 2017) devised new schemes with the previously unattained invisibility property. This property says that the set of admissible operations for the sanitizer remains hidden from outsiders. Subsequently, Beck *et al.* (ACISP 2017) gave an even stronger version of this notion and constructions achieving it. Here we characterize the invisibility property in both forms by showing that invisible sanitizable signatures are equivalent to IND−CPA-secure encryption schemes, and strongly invisible signatures are equivalent to IND−CCA2-secure encryption schemes. The equivalence is established by proving that invisible (resp. strongly invisible) sanitizable signature schemes yield IND−CPA-secure (resp. IND−CCA2-secure) public-key encryption schemes and that, vice versa, we can build (strongly) invisible sanitizable signatures given a corresponding public-key encryption scheme.

Keywords: Sanitizable signatures · Digital signatures · Invisibility
Public-key encryption · One-way functions

1 Introduction

Sanitizable signature schemes enable the signer of a document to declare certain sections of the message as admissible for modification, so that another designated party (the sanitizer) can modify them and update the signature without affecting the authenticity and integrity of the immutable parts. The main motivation is to balance out the verifier's wish to check authenticity of parts of the original document and the signer's desire for privacy of the sanitized data. The idea of sanitizable signature schemes dates back to a work by Ateniese *et al.* [2].

In [2], the authors introduced several security properties for sanitizable signature schemes. Besides unforgeability against outsiders, a desirable property is immutability, which demands that even a malicious sanitizer can only alter

B. Preneel and F. Vercauteren (Eds.): ACNS 2018, LNCS 10892, pp. 202–220, 2018.
https://doi.org/10.1007/978-3-319-93387-0_11

admissible parts. Privacy asks that one cannot reconstruct the original document given only the sanitized version and signature, and its strengthening called unlinkability [7] says that one cannot determine the origin to a sanitized document among several known possibilities. Signer and sanitizer accountability ensure that in case of a dispute the parties can give a convincing proof of who created a signature, the signer or the sanitizer. A less common property is transparency, which should hide who created a signature, in case neither of the parties helps to determine the originator—this stands in contrast to public accountability, where no additional help is required to determine who signed the document.

1.1 Invisible Sanitizable Signatures

Recently, Camenisch *et al.* [10] formalized the notion of invisibility of sanitizable signatures. This property, formerly called strong transparency in [2], should hide which modifications a sanitizer is allowed to perform. In previous constructions the description of admissible operations, denoted ADM, had usually been attached in clear to the signature. Gong *et al.* [25] were the first to argue that this information can be of value, and later Camenisch *et al.* showed that hiding it may be a desirable goal. They also revised the theoretical framework of sanitizable signatures in order to capture the invisibility property, and gave constructions achieving it based on a new type of chameleon hash functions with ephemeral trapdoors. Soon after, Beck *et al.* [3] further strengthened the notion of invisibility.

In its basic form, invisibility protects against leakage of ADM if the sanitizer public key is only used in connection with a single signer. In applications this means that the sanitizer must create a fresh key pair for each user. Strong invisibility, on the other hand, allows to use the same sanitizer key pair with multiple signers. Beck *et al.* use unique signatures, $\mathsf{IND-CCA2}$-secure encryption, and collision-resistant chameleon hash functions to achieve strong invisibility.

Technically, the difference between the two invisibility notions lies in the capabilities of an adversary trying to establish which of two potential operation sets, ADM_0 or ADM_1, has been encoded as admissible into the signature. Given a challenge signature, the adversary may query a sanitizing oracle on it as long as the requested modification does not allow to distinguish the two cases trivially (this happens e.g. if the modification is admissible for one of the two sets but not for the other). For the basic invisibility notion the adversary can ask for sanitizations only in connection with the public key $\mathsf{pk}_{\mathsf{Sig}}$ of the genuine signer. In the stronger notion, the adversary can also request sanitizations of messages signed with other, possibly maliciously chosen signer keys $\mathsf{pk}'_{\mathsf{Sig}}$.

1.2 Our Contributions

In this work we show that invisible sanitizable signature schemes and public-key encryption schemes are equivalent. Our equivalence proof consists of four parts.

Invisibility Implies IND−CPA*-Secure Encryption.* Our first result is to show that an invisible sanitizable signature scheme yields an IND−CPA-secure bit-encryption scheme. An invisible scheme hides the actual admissible operations for a signature; we can use this property to securely embed a message bit b by using one of two fixed and distinct admissible operation descriptions (ADM_0 or ADM_1) to build a signature σ under a fresh signer key pair. The ciphertext consists of the signature σ and the signer public key $\mathsf{pk}_{\mathsf{Sig}}$. Invisibility now guarantees that no outsider is able to distinguish the two cases.

The trapdoor information for decryption is the sanitizer secret key; his public key acts as the public key of the encryption scheme. With his secret key, the sanitizer can run the sanitization process and check via a distinguishing modification which operation ADM_b has been embedded: Only the admissible one (ADM_b) will result in a valid new signature. For the other operation (ADM_{1-b}), the modification should fail by the immutability property of the sanitizable scheme. Note that we obviously need some other security property besides invisibility, because it is easy to devise invisible signature schemes without any other security property, e.g. with constant signatures.

Strong Invisibility Implies IND−CCA2*-Secure Encryption.* While the construction of an IND−CPA-secure scheme via the embedding of the hidden ADM may be expected, we argue next that the same construction yields an IND−CCA2-secure encryption scheme if the underlying sanitizable signature scheme is strongly invisible. This result is less conventional, since it links the sanitization for different signer keys with the ability to securely decrypt different ciphertexts.

The proof idea is to note that ciphertexts in our encryption system are of the form $(\sigma, \mathsf{pk}_{\mathsf{Sig}})$. Given a challenge ciphertext $(\sigma, \mathsf{pk}_{\mathsf{Sig}})$, recall that for IND−CCA2-security we must allow for oracle decryptions of ciphertexts $(\sigma', \mathsf{pk}'_{\mathsf{Sig}}) \neq (\sigma, \mathsf{pk}_{\mathsf{Sig}})$. Since decryption is performed via sanitization, and strong invisibility allows us to call the sanitizer for different keys $\mathsf{pk}'_{\mathsf{Sig}}$, we can easily decrypt ciphertexts of the form $(\sigma', \mathsf{pk}'_{\mathsf{Sig}})$ with $\mathsf{pk}'_{\mathsf{Sig}} \neq \mathsf{pk}_{\mathsf{Sig}}$. To handle ciphertexts $(\sigma', \mathsf{pk}_{\mathsf{Sig}})$ under the original signer key we rely on the strong unforgeability property of the signature scheme: it says that one cannot create fresh signatures σ' under $\mathsf{pk}_{\mathsf{Sig}}$, and therefore an IND−CCA2-adversary cannot submit valid oracle queries of this form.

In a sense, this result warrants the deployment of an IND−CCA2-secure encryption scheme in the strongly invisible construction of Beck *et al.* [3]: Any strongly invisible sanitizable signature scheme already implies IND−CCA2-secure encryption systems. Note that we construct an IND−CCA2-secure *bit* encryption scheme, but this is sufficient to derive an IND−CCA2-secure *string* encryption scheme [14,26,31,32].

IND−CPA*-Secure Encryption Implies Invisibility.* Next we establish the converse implication, i.e. from IND−CPA-secure public-key encryption schemes to invisible sanitizable signatures. Note that the existence of the former primitive also implies the existence of one-way functions (the argument is identical to the one in [35, Lemma 1]), and thus of secure digital signature schemes [33,35], so that we can

use this building block in our construction as well. Besides invisibility, the derived sanitizable signature scheme has all the common properties, like unforgeablility, immutability, privacy, and accountability.

The construction idea is to have the signer sign every message block of the message with a different, ephemeral key, and then to protect this tuple of signatures with an additional signature under his secret key. This "message" part of the signature ensures unforgeability, privacy, and accountability. To enable the sanitizer to modify the admissible blocks, the signer appends another "administrative" signature over the description ADM and the tuple of secret keys used to sign the admissible blocks, both encrypted under the sanitizer public encryption key to allow for invisibility. If some admissible block has to be modified, the sanitizer can retrieve the corresponding ephemeral key via decryption, change the message block and then update the relevant signatures in the "message" part. Immutability (i.e., protection against inadmissible modifications from a malicious sanitizer) then follows from the unforgeability of the underlying digital signature scheme: It is ensured by the fact that the sanitizer only receives the signing keys for the blocks he is allowed to modify.

We stress here that our construction does not achieve some less common properties, in particular transparency and unlinkability, and that our security reduction is non-tight. On the other hand, we regard our work as being above all a feasibility result, so that tightness—even though desirable—should not be viewed as essential, and we believe that invisible, non-transparent sanitizable signatures can have interesting applications: One can envision scenarios where it should be impossible to learn which (if any) message blocks have the potential to be altered, but on the other hand it should be clear who signed the document (e.g., for legal and accountability reasons).

IND−CCA2-*Secure Encryption Implies Strong Invisibility.* The noteworthy property of the above construction is that IND−CPA-security suffices to achieve (ordinary) invisibility. With a slight technical twist, we interestingly achieve strong invisibility if we now have an IND−CCA2-secure encryption scheme: Namely, we include the signer public key in the encryption of ADM and the trapdoor information for the sanitizer. Hence, together with our converse construction of IND−CCA2-secure encryption from strong invisibility, we also characterize this form of invisibility through public-key encryption.

In light of the strongly invisible construction of Beck *et al.* [3], which besides an IND−CCA2-secure encryption scheme also relies on signature schemes and collision-resistant chameleon hash functions, our solution shows that the former (together with a regular signature scheme) suffices. Of course, the solution by Beck *et al.* is significantly more efficient.

1.3 Related Work

As mentioned above, sanitizable signature schemes were introduced by Ateniese *et al.* in their foundational work [2]. The first, and to this date widely adopted security model describing this primitive is due to Brzuska *et al.* [5], where the

authors formalized the unforgeability, immutability, privacy, transparency, and accountability properties of a sanitizable signature scheme with game-based security definitions. Later on, Brzuska *et al.* added the important unlinkability property [7,9], as well as non-interactive public accountability [8,9], to the picture of security notions—see Appendix C in the full version [21] for all the definitions.

Subsequently, the formal framework introduced in [5] came under scrutiny by Gong *et al.* [25], who pointed out that sanitizable signatures formalized as above were vulnerable to so-called rights-forge attacks. Their solution was to introduce stronger versions of unforgeability, immutability and accountability, which also consider the admissible blocks in the security experiments. Even stronger variants of unforgeability, privacy, transparency, and accountability were later provided by Krenn *et al.* [30], who decided to also track the signatures in the definitions (in much the same way as for regular signature schemes, when upgrading from "ordinary" to strong unforgeability). Finally, the invisibility property was formalized by Camenisch *et al.* [10], following ideas already discussed in [2], and recently further strengthened by Beck *et al.* [3].

The above literature deals with sanitizable signature schemes as they are intended here. On the other hand, we point out that there are many other primitives and extensions that are closely related to, but slightly different from sanitizable signature schemes as treated in this work. Among these there are redactable signatures [4,16,18,28], sanitizable signatures where sanitizer modifications are limited to certain values [11,19,29,34] or where the signer is allowed to add sanitizers after having signed the message [13,36], sanitizable signatures supporting a multi-signer, multi-sanitizer paradigm [6,9,12], or allowing for sanitization of signed, encrypted data [15,20]. More generally, we note that this whole body of literature falls under the broad category of computing on authenticated data [4,23,24]. We refer to the extensive overviews of Ahn *et al.* [1] and Demirel *et al.* [17] for further information.

We conclude the related work overview by mentioning that our work also continues a line of research started in [6], where the authors showed that it is possible to construct a sanitizable signature scheme achieving unforgeability, immutability, privacy, and accountability only assuming that arbitrary secure signature schemes exist, i.e. only assuming that one-way functions exist. In this regard, and in light of known separation results of public-key cryptography and one-wayness [27], our work proves that the same does most likely not hold for (strongly) invisible sanitizable signature schemes.

1.4 Organization

In Sect. 2 we outline the syntax of sanitizable signature schemes (and the corresponding specific notation), give an overview of the correctness and security notions, and discuss the invisibility property. In Sect. 3 we show how to construct a public-key bit-encryption scheme from an invisible sanitizable signature scheme, and we prove the corresponding security results, whereas Sect. 4 is devoted to the converse implication. Finally, we draw some conclusions in Sect. 5.

2 Definition of Sanitizable Signatures

2.1 Notation

The starting point of our theoretical discussion on sanitizable signatures is the security model introduced by Brzuska *et al.* in [5]. However, since invisibility will play a crucial role in our work, their framework has to be slightly adapted. Their approach often relies on the fact that the description ADM of admissible parts is recoverable from signatures, in direct contrast to the invisibility property which aims to hide this information. Thus, before we can actually start with the definition of sanitizable signatures, we need to introduce some preliminary notation. In doing so we mainly follow the work of Camenisch *et al.* [10].

Messages $m \in \mathcal{M}$ are assumed to consist of a finite number of *blocks*, each block being an element from a set $\mathcal{B}$ (usually $\mathcal{B} \subseteq \{0,1\}^*$). The message space $\mathcal{M}$ is thus a subset of $\mathcal{B}^*$. We use the notation $m[i]$ to refer to the i-th block and write $m = (m[1], \ldots, m[\ell])$ to stress that the message m consists of ℓ blocks.

Admissible blocks in a message $m = (m[1], \ldots, m[\ell]) \in \mathcal{M}$ are identified by means of the parameter $\mathrm{ADM} = (A, l) \in \mathcal{P}(\mathbb{N}) \times \mathbb{N}$ (also called *sanitizing rights*), where $l \in \mathbb{N}$ denotes the total number of blocks a message must have, while $A := \{a_1, \ldots, a_j\}$ is the set containing the indices of the blocks the sanitizer is allowed to modify. Of course, here we need $1 \leq a_1, \ldots, a_j \leq l$, a condition that we will always assume to be satisfied. We then say that ADM *matches* m if $\ell = l$, in which case we write $\mathrm{ADM}(m) = \top$ (otherwise $\mathrm{ADM}(m) = \bot$). If $\mathrm{ADM}_0 = (A_0, l)$ and $\mathrm{ADM}_1 = (A_1, l)$ are two sanitizing rights matching m, we define $\mathrm{ADM}_0 \cap \mathrm{ADM}_1 := (A_0 \cap A_1, l)$. Similarly, to identify admissible block indices, we write $a \in \mathrm{ADM} = (A, l)$ if $1 \leq a \leq l$ and $a \in A$.

If $m = (m[1], \ldots, m[\ell]) \in \mathcal{M}$ is a message, the actual modifications to certain blocks made by the sanitizer (i.e., the *sanitizing instructions*) are identified by means of the parameter $\mathrm{MOD} = (M, l) \in \mathcal{P}(\mathbb{N} \times \mathcal{B}) \times \mathbb{N}$, where $l \in \mathbb{N}$ denotes the total number of blocks in a message and $M := \{(i_1, \bar{m}_1), \ldots, (i_k, \bar{m}_k)\}$ denotes the set of changes made by the sanitizer. Here $(i, \bar{m}) \in M$ is intended to mean that the sanitizer will replace block $m[i]$ with $\bar{m}$. Again, here we need $1 \leq i_1, \ldots, i_k \leq l$, which we will assume throughout. We then say that MOD *matches* m if $\ell = l$, in which case we write $\mathrm{MOD}(m)$ for the message m' obtained by modifying m according to MOD, i.e. $m' = \mathrm{MOD}(m)$ if and only if $m' = (m'[1], \ldots m'[\ell]) \in \mathcal{M}$ and, for every $1 \leq i \leq \ell$, $m'[i] = \bar{m}_i$ if $i \in \{i_1, \ldots, i_k\}$, and $m'[i] = m[i]$ otherwise. We write $\mathrm{MOD}(m) = \bot$ if MOD does not match m.

Finally, recall that the sanitizer is supposed to modify only message blocks declared as admissible by the signer. In this regard, the following notation will be useful: If $\mathrm{ADM} = (A, l_{\mathrm{ADM}})$ and $\mathrm{MOD} = (M, l_{\mathrm{MOD}})$ are as above, we say that MOD *matches* (or *is valid w.r.t.*) ADM if $l_{\mathrm{ADM}} = l_{\mathrm{MOD}}$ and $\widetilde{M} \subseteq A$, where $\widetilde{M} := \{i_1, \ldots, i_k\}$ is the set of indices of the blocks which the sanitizer intends to modify (as specified by M). In this case we write $\mathrm{MOD}(\mathrm{ADM}) = \top$, otherwise $\mathrm{MOD}(\mathrm{ADM}) = \bot$.

2.2 Definition of Sanitizable Signature Schemes

With the notation introduced above we are now ready to define sanitizable signature schemes. The definition is based on the one given by Brzuska *et al.* in [5] but takes into account that the sanitizing rights are no longer publicly recoverable from a valid message-signature pair. We remark here that, nonetheless, the sanitizer is always able to learn which message blocks he can modify by trying to sanitize them singularly and checking if the resulting signature is valid, an operation linear in the number of blocks of the message. This is the reason why we do not include ADM as an additional input to the Sanit algorithm: Either it is implicitly in the signatures or it must be communicated out-of-band.

Since our definition is similar to the one in [10], we give here only a schematic overview of the algorithms comprising a sanitizable signature scheme and their syntax. The complete definition can be found in the full version [21].

Definition 1. *A* sanitizable signature scheme SSS *is a tuple of eight probabilistic polynomial-time algorithms* $\mathsf{SSS} := (\mathsf{PGen}, \mathsf{KGen}_{\mathsf{Sig}}, \mathsf{KGen}_{\mathsf{San}}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$*, whose syntax is as follows:*

- $\mathsf{pp} \leftarrow_{\$} \mathsf{PGen}(1^{\lambda})$*, to generate public parameters;*
- $(\mathsf{pk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow_{\$} \mathsf{KGen}_{\mathsf{Sig}}(\mathsf{pp})$*, to generate signing keys;*
- $(\mathsf{pk}_{\mathsf{San}}, \mathsf{sk}_{\mathsf{San}}) \leftarrow_{\$} \mathsf{KGen}_{\mathsf{San}}(\mathsf{pp})$*, to generate sanitization keys;*
- $\sigma \leftarrow_{\$} \mathsf{Sign}(\mathsf{pp}, m, \mathsf{sk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}}, \mathrm{ADM})$*, for signatures;*
- $\sigma' \leftarrow_{\$} \mathsf{Sanit}(\mathsf{pp}, m, \sigma, \mathsf{sk}_{\mathsf{San}}, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}}, \mathrm{MOD})$*, for sanitized signatures;*
- $d \leftarrow \mathsf{Verify}(\mathsf{pp}, m, \sigma, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$*, for verification;*
- $\pi \leftarrow_{\$} \mathsf{Proof}(\mathsf{pp}, m, \sigma, \{(m_i, \sigma_i)\}_{i=1}^{k}, \mathsf{sk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$*, to generate proofs;*
- $d \leftarrow \mathsf{Judge}(\mathsf{pp}, m, \sigma, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}}, \pi)$*, to determine who signed the document.*

2.3 Correctness and Security Properties of Sanitizable Signature Schemes

We now turn to the definition of correctness and the statement of security properties of a sanitizable signature scheme SSS. As for correctness, we follow Brzuska *et al.* [5] and subsequent work and require that the following three properties hold. We give only an informal description here and refer to Appendix B in the full version [21] for complete definitions, as adapted to our framework.

- *Signing Correctness*: Every time an honest signer signs a message $m \in \mathcal{M}$ with sanitizing rights matching m, he produces a valid signature $\sigma \neq \perp$ such that (m, σ) verifies under the corresponding public keys;
- *Sanitizing Correctness*: Every time the intended sanitizer honestly sanitizes a valid message-signature pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$ with sanitizing instructions MOD matching the sanitizing rights given to him by the signer, he produces a valid signature $\sigma' \neq \perp$ such that $(\mathrm{MOD}(m), \sigma')$ verifies under the corresponding public keys;
- *Proof Correctness*: Every time an honest signer generates a proof regarding a valid message-signature pair, Judge identifies the correct accountable party.

Next we discuss the relevant security properties of a sanitizable signature scheme SSS. Most of these properties were introduced in their basic form by Brzuska *et al.* in [5] and later in [7,8]. We will be mainly concerned with their "strong" counterparts as formalized by Krenn *et al.* in [30] and later adopted by Camenisch *et al.* [10] and Beck *et al.* [3]. The definitions we adopt take into account that the sanitizing rights ADM (which are no longer assumed to be publicly recoverable from a valid message-signature pair) are an information which needs protection, as work by Gong *et al.* [25] has shown. In particular, by requiring a sanitizable signature scheme to satisfy the "strong" versions of the unforgeability, signer- and sanitizer-accountability properties, we mostly avoid so-called *rights forge attacks* as discussed in [25] (for immutability the matter is more delicate—see Appendix C in the full version [21] for further discussions).

We again give only a brief and intuitive description of the security properties here and refer the interested reader to Appendix C in the full version [21] for complete definitions and the corresponding security experiments. Only the notion of invisibility, central to our work, will be discussed here in detail.

- *Unforgeability*: No adversary should be able to produce a valid message-signature pair never seen before;
- *Immutability*: The sanitizer should be able to modify only message blocks previously declared as admissible by the signer;
- *Privacy*: Given a valid, sanitized message-signature pair, no adversary should be able to recover any information about the original content of the sanitized blocks;
- *Transparency*: Given a valid message-signature pair, no adversary should be able to determine whether it was the signer or the sanitizer who produced the signature;
- *Signer-Accountability*: A malicious signer should not be able to produce a valid message-signature pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$ and a proof which induces Judge into erroneously blaming the sanitizer for (m, σ);
- *Sanitizer-Accountability*: A malicious sanitizer should not be able to produce a valid message-signature pair $(m', \sigma') \in \mathcal{M} \times \mathcal{S}$ such that legitimate proofs generated by the signer induce Judge into blaming the signer for (m', σ');
- *Unlinkability*: Given a valid message-signature pair $(m', \sigma') \in \mathcal{M} \times \mathcal{S}$ that has been sanitized, no adversary should be able to decide from which known pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$ it originated from;
- *Non-Interactive Public Accountability*: The party accountable for a valid message-signature pair can be determined publicly, without the need of any further information. In particular, the Proof algorithm is trivial.

2.4 (Strong) Invisibility

Loosely speaking, a sanitizable signature scheme is *invisible* if, given a valid message-signature pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$, no adversary is able to decide if any specific message block is admissible (i.e., can be modified by the sanitizer) or immutable. This property was first introduced by Ateniese *et al.* in their foundational work [2] under the name "strong transparency" (an expression later fallen

into disuse, not to be confused with the notion of transparency defined in the literature). However, they did not provide any formal definition or construction achieving it. It was later abandoned by Brzuska *et al.* [5] on the grounds that it appeared to be too strong. Indeed, since they worked under the assumption that ADM is always publicly recoverable from a valid signature $\sigma \neq \perp$ (in obvious conflict with the invisibility notion), it was in fact unachievable. Later on, the invisibility property was considered by Camenisch *et al.* [10], who defined it formally and gave the first provably secure construction of an invisible sanitizable signature scheme. A stronger version of invisibility was later defined by Beck *et al.* in [3], where the sanitizer may use his public key with multiple signers.

In the invisibility security experiment, a signer and a sanitizer key pair are generated and a bit $b \leftarrow_{\$} \{0,1\}$ is chosen uniformly at random and kept secret. An adversary $\mathcal{A}$ is given access to an oracle $\mathcal{O}^{\mathsf{LoR}}$ which, on input a message and two sanitizing rights ADM_0, ADM_1, produces a signature σ (under the signer secret key and the sanitizer public key) making ADM_b admissible. In addition, $\mathcal{A}$ has adaptive access to restricted signing, sanitizing, and proof oracles.

We remark that, in the above experiment, a restricted signing oracle (with fixed sanitizer public key $\mathsf{pk}_{\mathsf{San}}$) can be simulated by querying $\mathcal{O}^{\mathsf{LoR}}$ and putting $\mathrm{ADM}_0 = \mathrm{ADM}_1$. Furthermore, for sanitization requests of any message-signature pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$ with $\sigma \leftarrow_{\$} \mathcal{O}^{\mathsf{LoR}}(m, \mathrm{ADM}_0, \mathrm{ADM}_1)$, $\mathcal{A}$ must be limited to modifications matching $\mathrm{ADM}_0 \cap \mathrm{ADM}_1$ in order to avoid trivial attacks exposing b. This is why all queries to and answers from $\mathcal{O}^{\mathsf{LoR}}$, together with the allowed sanitizing rights $\mathrm{ADM}_0 \cap \mathrm{ADM}_1$, are recorded in a "whitelist" W: Whenever $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{Sanit}}$, the oracle goes through the list W of previously signed messages, to see which blocks the adversary is indeed allowed to modify. If the query is accepted, the whitelist has to be updated to also include the new (sanitized) message-signature pair, with the same sanitizing rights as the original pair (this has to be done because a sanitized message could be sanitized again). In the basic invisibility property the answers are only computed for the given key $\mathsf{pk}_{\mathsf{Sig}}$.

The adversary's goal is to guess b, i.e., to decide which set of blocks the oracle $\mathcal{O}^{\mathsf{LoR}}$ has made admissible. The scheme SSS is invisible if no efficient adversary as above succeeds in doing so with probability significantly greater than $1/2$.

We observe that the definition of invisibility already has the flavor of the "strong" variant of the definitions given in [10,30], in that we always keep track of the signatures in the whitelist W. On the other hand, the main drawback of this definition is that it is not possible to query the sanitization oracle for keys different from the challenge ones. As remarked by Beck *et al.* [3], this may have undesirable consequences: $\mathcal{A}$ could pose as another signer and, as soon as he gets access to a sanitization oracle, could potentially learn the bit b.

To address these concerns (and to give a definition of invisibility that also protects against dishonest signers), one can allow queries to the sanitization oracle with public keys chosen by the adversary $\mathcal{A}$. This approach leads to the definition of strong invisibility. The main difference between the invisibility and the strong invisibility experiments is that the adversary is allowed oracle queries to $\tilde{\mathcal{O}}^{\mathsf{LoR}}$ and $\tilde{\mathcal{O}}^{\mathsf{Sanit}}$ with adversarially chosen public keys. A sanitizable signature scheme

secure in this stronger sense does not suffer from the flaw mentioned above. As a side effect, the signing oracle derived from $\tilde{\mathcal{O}}^{\mathsf{LoR}}$ is no longer restricted. The formal definition of (strong) invisibility is given in the full version [21].

3 Invisible Sanitizable Signatures Imply Public-Key Encryption Schemes

In this section we show how to construct a public-key bit-encryption scheme from an invisible sanitizable signature scheme.

3.1 Construction

Suppose that Alice wants to send a secret bit $b \in \{0,1\}$ to Bob, without an adversary $\mathcal{A}$ being able to learn it. To do so, Bob publicly chooses a sanitizable signature scheme SSS and a security parameter $\lambda \in \mathbb{N}$, and generates a tuple of public parameters $\mathsf{pp} \leftarrow_{\$} \mathsf{PGen}(1^{\lambda})$. Observe that the block set $\mathcal{B}$ defined by pp clearly must contain at least two elements—we will assume that $\{0,1\} \subseteq \mathcal{B}$, but for other block sets the adjustment is straightforward. Moreover, we assume that the two-block-messages $(0,0),(1,0),(0,1)$ belong to the message space $\mathcal{M}$, but again our construction can be easily modified should this not be the case.

Bob then generates a sanitizer key pair $(\mathsf{pk}_{\mathsf{San}}, \mathsf{sk}_{\mathsf{San}}) \leftarrow_{\$} \mathsf{KGen}_{\mathsf{San}}(\mathsf{pp})$, and chooses a message $m \in \mathcal{M}$ consisting of two blocks, e.g. $m = (0,0)$. He sends $(\mathsf{pp}, m, \mathsf{pk}_{\mathsf{San}})$ to Alice over an unprotected channel, while $\mathsf{sk}_{\mathsf{San}}$ is kept secret.

Upon receiving $(\mathsf{pp}, m, \mathsf{pk}_{\mathsf{San}})$, Alice runs $(\mathsf{pk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow_{\$} \mathsf{KGen}_{\mathsf{Sig}}(\mathsf{pp})$ to generate a signer key pair. Now, depending on whether she wants to encrypt $b = 0$ or $b = 1$, she signs the message m declaring as admissible the first or the second block, respectively. She then sends the signature σ and her public key $\mathsf{pk}_{\mathsf{Sig}}$ to Bob, while $\mathsf{sk}_{\mathsf{Sig}}$ is kept secret.

Upon receiving $(\sigma, \mathsf{pk}_{\mathsf{Sig}})$, Bob tries to separately modify the first and the second message block by replacing it with '1'. He thus sets $\mathrm{MOD}_0 \leftarrow (\{(1,1)\},2)$ and $\mathrm{MOD}_1 \leftarrow (\{(2,1)\},2)$ and then computes $\sigma'_0 \leftarrow_{\$} \mathsf{Sanit}(\mathsf{pp}, m, \sigma, \mathsf{sk}_{\mathsf{San}}, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}}, \mathrm{MOD}_0)$ and $\sigma'_1 \leftarrow_{\$} \mathsf{Sanit}(\mathsf{pp}, m, \sigma, \mathsf{sk}_{\mathsf{San}}, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}}, \mathrm{MOD}_1)$.

Now, assuming that SSS is sanitizing correct and immutable, exactly one of the two signatures computed by Bob will be valid. If Alice has encrypted $b = 0$, then σ'_0 will be valid with overwhelming probability (because of the sanitizing correctness property), while σ'_1 will be either invalid or equal to $\perp$ with very high probability (because SSS is immutable). On the other hand, if Alice has chosen $b = 1$, then σ'_1 will be valid and σ'_0 not by the same argument. In the unlikely event that both signatures are valid or both are invalid, Bob cannot decrypt the message sent by Alice.

We thus conclude that Bob is able to correctly decrypt the bit encrypted by Alice with very high probability by sanitizing m twice and checking the signatures (or error messages). Moreover, if we also assume SSS to be invisible, then any adversary $\mathcal{A}$ will be able to learn b only with negligible probability. In fact, from an outsider's perspective learning b is equivalent to establishing

which message block is admissible, which is highly unlikely by the invisibility assumption.

We now turn to a more rigorous definition of our public-key bit-encryption scheme, as well as to the statement of the correctness and security properties.

Construction 1. *Let* $\mathsf{SSS} := (\mathsf{PGen}, \mathsf{KGen}_{\mathsf{Sig}}, \mathsf{KGen}_{\mathsf{San}}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$ *be a sanitizable signature scheme. We define a public-key bit-encryption scheme* Π *as in Fig. 1.*

$\Pi.\mathsf{PGen}(1^\lambda)$:

1 $\mathsf{pp}_{\mathsf{SSS}} \leftarrow_\$ \mathsf{SSS.PGen}(1^\lambda)$
2 $\mathcal{M}_\Pi \leftarrow \{0,1\}$, $\mathcal{C}_\Pi \leftarrow \mathcal{S}_{\mathsf{Sig}} \times \mathcal{K}_{\mathsf{Sig,pk}}$, $\mathcal{K}_\Pi \leftarrow \mathcal{K}_{\mathsf{San}}$
3 $m \leftarrow (0,0)$
4 $\mathsf{pp}_\Pi \leftarrow (\mathsf{pp}_{\mathsf{SSS}}, \mathcal{M}_\Pi, \mathcal{C}_\Pi, \mathcal{K}_\Pi, m)$
5 return pp_Π

$\Pi.\mathsf{KGen}(\mathsf{pp}_\Pi)$:

1 parse $\mathsf{pp}_\Pi = (\mathsf{pp}_{\mathsf{SSS}}, \mathcal{M}_\Pi, \mathcal{C}_\Pi, \mathcal{K}_\Pi, m)$
2 $(\mathsf{pk}_{\mathsf{San}}, \mathsf{sk}_{\mathsf{San}}) \leftarrow_\$ \mathsf{SSS.KGen}_{\mathsf{San}}(\mathsf{pp}_{\mathsf{SSS}})$
3 $\mathsf{pk}_\Pi \leftarrow \mathsf{pk}_{\mathsf{San}}$, $\mathsf{sk}_\Pi \leftarrow \mathsf{sk}_{\mathsf{San}}$
4 return $(\mathsf{pk}_\Pi, \mathsf{sk}_\Pi)$

$\Pi.\mathsf{Enc}(\mathsf{pp}_\Pi, b, \mathsf{pk}_\Pi)$:

1 parse $\mathsf{pp}_\Pi = (\mathsf{pp}_{\mathsf{SSS}}, \mathcal{M}_\Pi, \mathcal{C}_\Pi, \mathcal{K}_\Pi, m)$, $\mathsf{pk}_\Pi = \mathsf{pk}_{\mathsf{San}}$
2 $(\mathsf{pk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow_\$ \mathsf{SSS.KGen}_{\mathsf{Sig}}(\mathsf{pp}_{\mathsf{SSS}})$
3 $\mathrm{ADM}_0 \leftarrow (\{1\}, 2)$
4 $\mathrm{ADM}_1 \leftarrow (\{2\}, 2)$
5 $\sigma \leftarrow_\$ \mathsf{SSS.Sign}(\mathsf{pp}_{\mathsf{SSS}}, m, \mathsf{sk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}}, \mathrm{ADM}_b)$
6 return $(\sigma, \mathsf{pk}_{\mathsf{Sig}})$

$\Pi.\mathsf{Dec}(\mathsf{pp}_\Pi, c, \mathsf{pk}_\Pi, \mathsf{sk}_\Pi)$:

1 parse $\mathsf{pp}_\Pi = (\mathsf{pp}_{\mathsf{SSS}}, \mathcal{M}_\Pi, \mathcal{C}_\Pi, \mathcal{K}_\Pi, m)$, $c = (\sigma, \mathsf{pk}_{\mathsf{Sig}})$, $\mathsf{pk}_\Pi = \mathsf{pk}_{\mathsf{San}}$, $\mathsf{sk}_\Pi = \mathsf{sk}_{\mathsf{San}}$
2 $d \leftarrow \mathsf{SSS.Verify}(\mathsf{pp}_{\mathsf{SSS}}, m, \sigma, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$
3 if $d = \bot$ then
4 return $\bot$
5 $\mathrm{MOD}_0 \leftarrow (\{(1,1)\}, 2)$
6 $\mathrm{MOD}_1 \leftarrow (\{(2,1)\}, 2)$
7 $m'_0 \leftarrow \mathrm{MOD}_0(m)$, $m'_1 \leftarrow \mathrm{MOD}_1(m)$
8 $\sigma'_0 \leftarrow_\$ \mathsf{SSS.Sanit}(\mathsf{pp}_{\mathsf{SSS}}, m, \sigma, \mathsf{sk}_{\mathsf{San}}, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}}, \mathrm{MOD}_0)$
9 $\sigma'_1 \leftarrow_\$ \mathsf{SSS.Sanit}(\mathsf{pp}_{\mathsf{SSS}}, m, \sigma, \mathsf{sk}_{\mathsf{San}}, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}}, \mathrm{MOD}_1)$
10 $d_0 \leftarrow \mathsf{SSS.Verify}(\mathsf{pp}_{\mathsf{SSS}}, m'_0, \sigma'_0, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$
11 $d_1 \leftarrow \mathsf{SSS.Verify}(\mathsf{pp}_{\mathsf{SSS}}, m'_1, \sigma'_1, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$
12 if $d_0 = \top \ \wedge \ d_1 = \bot$ then
13 return 0
14 if $d_0 = \bot \ \wedge \ d_1 = \top$ then
15 return 1
16 return $\bot$

Fig. 1. Public-key bit-encryption scheme from an invisible sanitizable signature scheme

3.2 IND−CPA-Security

We now formally state our security results about the public-key bit encryption scheme in Construction 1.

Theorem 2. *Let* $\mathsf{SSS} := (\mathsf{PGen}, \mathsf{KGen}_{\mathsf{Sig}}, \mathsf{KGen}_{\mathsf{San}}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$ *be a sanitizable signature scheme, and let* $\Pi := (\mathsf{PGen}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *be the public-key bit-encryption scheme defined in Construction 1. If* SSS *is sanitizing correct, immutable and invisible, then* Π *is correct and* IND−CPA*-secure.*

The proof gives a tight reduction in terms of the advantages: For any adversary $\mathcal{A}$ playing the IND−CPA-game we construct an adversary $\mathcal{B}$ against the invisibility game with roughly the same running time as $\mathcal{A}$, such that

$$\mathbf{Adv}^{\mathsf{IND-CPA}}_{\mathcal{A},\Pi}(\lambda) = \mathbf{Adv}^{\mathsf{Inv}}_{\mathcal{B},\mathsf{SSS}}(\lambda).$$

Note that we need the immutability property only to bound the correctness error by $2 \cdot \mathbf{Adv}^{\mathsf{Imm}}_{\mathcal{C},\mathsf{SSS}}(\lambda)$ for some efficient adversary $\mathcal{C}$ against the immutability game. The proof of Theorem 2 can be found in the full version [21].

3.3 IND−CCA2-Security

We next argue that the scheme above achieves IND−CCA2-security if SSS is assumed to be strongly invisible. Recall that the difference to regular invisibility is that now the adversary against strong invisibility can also make left-or-right signature requests for $(m, \mathsf{pk}'_{\mathsf{San}}, \mathrm{ADM}_0, \mathrm{ADM}_1)$ with different sanitizer public keys $\mathsf{pk}'_{\mathsf{San}} \neq \mathsf{pk}_{\mathsf{San}}$, and sanitization requests for $(m, \sigma, \mathsf{pk}'_{\mathsf{Sig}}, \mathrm{MOD})$ with different signer public keys $\mathsf{pk}'_{\mathsf{Sig}} \neq \mathsf{pk}_{\mathsf{Sig}}$. Interestingly, for our construction and proof we only rely on the latter property.

For the security proof we also need strong unforgeability of the sanitizable signature scheme. The reason is that ciphertexts are of the form $(\sigma, \mathsf{pk}_{\mathsf{Sig}})$, and the IND−CCA2-adversary may ask for decryptions of the form $(\sigma', \mathsf{pk}_{\mathsf{Sig}})$ where it alters the signature component for the same message. This would allow to break the security of the encryption scheme easily.

Theorem 3. *Let* $\mathsf{SSS} := (\mathsf{PGen}, \mathsf{KGen}_{\mathsf{Sig}}, \mathsf{KGen}_{\mathsf{San}}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$ *be a sanitizable signature scheme, and let* $\Pi := (\mathsf{PGen}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *be the public-key bit-encryption scheme defined in Construction 1. If* SSS *is sanitizing correct, strongly unforgeable, immutable and strongly invisible, then* Π *is correct and* IND−CCA2*-secure.*

The proof also gives a tight reduction in terms of the advantages: For any adversary $\mathcal{A}$ playing the IND−CCA2-game we construct adversaries $\mathcal{B}$ and $\mathcal{C}$ with roughly the same running time as $\mathcal{A}$, such that

$$\mathbf{Adv}^{\mathsf{IND-CCA2}}_{\mathcal{A},\Pi}(\lambda) \leq \mathbf{Adv}^{\mathsf{SInv}}_{\mathcal{B},\mathsf{SSS}}(\lambda) + 2 \cdot \mathbf{Adv}^{\mathsf{SUnf}}_{\mathcal{C},\mathsf{SSS}}(\lambda).$$

In fact, for IND−CCA1-security regular unforgeability is sufficient. Once more, we need immutability only to bound the correctness error. The proof of Theorem 3 can be found in the full version [21].

4 Public-Key Encryption Implies Invisible Sanitizable Signatures

In this section we present our construction of an invisible sanitizable signature scheme, starting from a secure public-key encryption scheme.

4.1 Construction

Our construction based on public-key encryption follows the established encode-and-sign paradigm and exploits the idea of using chameleon hash functions and signing the hash values with a regular signature scheme Σ (see, e.g., [2,5]). The sanitizer can then find collisions for the hashes with the help of his trapdoor key, allowing him to modify the message. Here, instead of chameleon hashes we use the signature scheme Σ itself.

In our scheme, signatures consist of two parts: the "message" part ensures the basic unforgeability and accountability properties, and can be created by either of the two parties. In contrast, the "administrative" part contains the information needed by the sanitizer to perform modifications, and can be created only by the signer. Parts of the administrative information are encrypted under an encryption scheme Π under the sanitizer's public key, to ensure invisibility.

To begin with, the signer generates a key pair $(\mathsf{pk}_\Sigma, \mathsf{sk}_\Sigma)$ for Σ, while the sanitizer creates keys $(\mathsf{pk}'_\Sigma, \mathsf{sk}'_\Sigma)$ and $(\mathsf{pk}_\Pi, \mathsf{sk}_\Pi)$ for Σ and Π, respectively. To sign a message $m = (m[1], \ldots, m[\ell])$, the signer generates a new key pair $(\mathsf{pk}^i_\Sigma, \mathsf{sk}^i_\Sigma)$ $(1 \le i \le \ell)$ for each block of m, signs every block with the corresponding key, and creates a tuple of signatures $S := (\sigma^1, \ldots, \sigma^\ell)$. He then generates a signature σ_{MSG} of the message $(0, m, S, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$ under sk_Σ. Here, m and S are signed so that they are protected from modification by outsiders, whereas $\mathsf{pk}_{\mathsf{Sig}}$, $\mathsf{pk}_{\mathsf{San}}$ and the initial bit '0' are included for technical reasons (namely, domain separation). The "message" part of the final signature σ then consists of (S, σ_{MSG}).

The first part of the signature must now be complemented with the information required to sanitize the admissible parts of the message, and to verify the signature. To this end, the signer generates the tuple $K_{\text{ADM}} = (\mathsf{sk}^{i_1}_\Sigma, \mathsf{sk}^{i_2}_\Sigma, \ldots)$ containing the secret keys of the admissible blocks $i_j \in \text{ADM}$ (properly padded to ensure a length-invariant encoding), and encloses it for the sanitizing party via encryption under pk_Π. In addition, we also hide the parameter ADM (to ensure invisibility) and the signer public key (in foresight of the strongly invisible version of our result) in this encryption. In summary, the signer creates an encryption C of $(\mathsf{pk}_{\mathsf{San}}, K_{\text{ADM}}, \text{ADM})$ under pk_Π and then, in order to prevent changes in these administrative data, creates a (regular) signature σ_{ADM} of the message $(1, \mathsf{pk}_{\mathsf{San}}, V, C)$. Here, $V := (\mathsf{pk}^1_\Sigma, \ldots, \mathsf{pk}^l_\Sigma)$ contains the verification keys for the single blocks, and again the initial bit '1' is included for domain separation reasons. The "administrative" part of the signature is then $(V, C, \sigma_{\text{ADM}})$, and the final signature is $\sigma := (S, \sigma_{\text{MSG}}, V, C, \sigma_{\text{ADM}})$.

If the sanitizer receives a signature σ for a message m, he first checks the validity of the signatures S, σ_{MSG} and σ_{ADM}, and recovers ADM and the corresponding signing keys K_{ADM} by decrypting C. Then, given valid sanitizing instructions $m' = \text{MOD}(m)$, he can update the "message" part of σ, leaving the "administrative" part unchanged. He obtains S' by substituting the relevant entries in S with new signatures of the modified blocks under the corresponding keys K_{ADM}, and updates σ'_{MSG} by re-signing $(0, m', S', \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$ under sk'_Σ. Finally, the sanitized signature for m' is given by $\sigma' = (S', \sigma'_{\text{MSG}}, V, C, \sigma_{\text{ADM}})$.

Immutability of the scheme is achieved by the fact that the sanitizer does not know the secret keys for the blocks he is not supposed to modify, and therefore cannot obtain suitable replacements for signatures in S. Observe that the signature σ_{MSG} immediately ensures public accountability, since it serves as a proof of who put the overall signature. This also implies that our scheme does not achieve transparency. For technical reasons it neither supports unlinkability.

Remark 1. The above discussion presumes that some mild assumptions on Σ and Π are satisfied, which we will henceforth assume to be in place. In particular, all signature keys must be of fixed length L (this can be achieved via padding of the keys), and the message blocks, as well as the tuples of the form $(0, m, S, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$ and $(1, \mathsf{pk}_{\mathsf{San}}, V, C)$, must lie in the message space of Σ (this is no restriction, because the signatures constructed in [33,35] support messages of arbitrary polynomial length). Also, ADM must be encoded in a length-invariant manner, and tuples of the form $(\mathsf{pk}_{\mathsf{San}}, K_{\mathrm{ADM}}, \mathrm{ADM})$ must lie in the message space of Π (which can be achieved through hybrid encryption).

We now turn to a more rigorous definition of our sanitizable signature scheme, as well as to the statement of the correctness and security results.

$\mathsf{SSS.PGen}(1^\lambda)$:

1 $\mathsf{pp}_\Pi \leftarrow_\$ \Pi.\mathsf{PGen}(1^\lambda)$
2 $\mathsf{pp}_\Sigma \leftarrow_\$ \Sigma.\mathsf{PGen}(1^\lambda)$
3 $\mathcal{M} \leftarrow \mathcal{M}_\Sigma^l$
4 $\mathsf{pp}_{\mathsf{SSS}} \leftarrow (\mathsf{pp}_\Pi, \mathsf{pp}_\Sigma, \mathcal{M})$
5 return $\mathsf{pp}_{\mathsf{SSS}}$

$\mathsf{SSS.KGen}_{\mathsf{Sig}}(\mathsf{pp}_{\mathsf{SSS}})$:

1 parse $\mathsf{pp}_{\mathsf{SSS}} = (\mathsf{pp}_\Pi, \mathsf{pp}_\Sigma, \mathcal{M})$
2 $(\mathsf{pk}_\Sigma, \mathsf{sk}_\Sigma) \leftarrow_\$ \Sigma.\mathsf{KGen}(\mathsf{pp}_\Sigma)$
3 $\mathsf{pk}_{\mathsf{Sig}} \leftarrow \mathsf{pk}_\Sigma$, $\mathsf{sk}_{\mathsf{Sig}} \leftarrow \mathsf{sk}_\Sigma$
4 return $(\mathsf{pk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}})$

$\mathsf{SSS.KGen}_{\mathsf{San}}(\mathsf{pp}_{\mathsf{SSS}})$:

1 parse $\mathsf{pp}_{\mathsf{SSS}} = (\mathsf{pp}_\Pi, \mathsf{pp}_\Sigma, \mathcal{M})$
2 $(\mathsf{pk}_\Pi, \mathsf{sk}_\Pi) \leftarrow_\$ \Pi.\mathsf{KGen}(\mathsf{pp}_\Pi)$
3 $(\mathsf{pk}'_\Sigma, \mathsf{sk}'_\Sigma) \leftarrow_\$ \Sigma.\mathsf{KGen}(\mathsf{pp}_\Sigma)$
4 $\mathsf{pk}_{\mathsf{San}} \leftarrow (\mathsf{pk}_\Pi, \mathsf{pk}'_\Sigma)$
5 $\mathsf{sk}_{\mathsf{San}} \leftarrow (\mathsf{sk}_\Pi, \mathsf{sk}'_\Sigma)$
6 return $(\mathsf{pk}_{\mathsf{San}}, \mathsf{sk}_{\mathsf{San}})$

$\mathsf{SSS.Sign}(\mathsf{pp}_{\mathsf{SSS}}, m, \mathsf{sk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}}, \mathrm{ADM})$:

1 if $\mathrm{ADM}(m) = \perp$ then return $\perp$
2 parse $\mathsf{pp}_{\mathsf{SSS}} = (\mathsf{pp}_\Pi, \mathsf{pp}_\Sigma, \mathcal{M})$, $\mathsf{sk}_{\mathsf{Sig}} = \mathsf{sk}_\Sigma$,
$m = (m[1], \ldots, m[l])$, $\mathsf{pk}_{\mathsf{Sig}} = \mathsf{pk}_\Sigma$,
$\mathsf{pk}_{\mathsf{San}} = (\mathsf{pk}_\Pi, \mathsf{pk}'_\Sigma)$, $\mathrm{ADM} = (A, l)$
3 $V \leftarrow \emptyset$, $S \leftarrow \emptyset$, $K_{\mathrm{ADM}} \leftarrow \emptyset$
4 for $1 \le i \le l$ do
5 $(\mathsf{pk}_\Sigma^i, \mathsf{sk}_\Sigma^i) \leftarrow_\$ \Sigma.\mathsf{KGen}(\mathsf{pp}_\Sigma)$
6 $V \leftarrow V \cup \{(i, \mathsf{pk}_\Sigma^i)\}$
7 $\sigma^i \leftarrow_\$ \Sigma.\mathsf{Sign}(\mathsf{pp}_\Sigma, m[i], \mathsf{sk}_\Sigma^i, \mathsf{pk}_\Sigma^i)$
8 $S \leftarrow S \cup \{(i, \sigma^i)\}$
9 $K_{\mathrm{ADM}} \leftarrow K_{\mathrm{ADM}} \cup \begin{cases} \{(i, \mathsf{sk}_\Sigma^i)\} & i \in A \\ \{(i, 0^L)\} & \text{else} \end{cases}$
10 parse $V = (\mathsf{pk}_\Sigma^1, \ldots, \mathsf{pk}_\Sigma^l)$, $S = (\sigma^1, \ldots, \sigma^l)$
$K_{\mathrm{ADM}} = (K_{\mathrm{ADM}}^1, \ldots, K_{\mathrm{ADM}}^l)$
11 $t \leftarrow (0, m, S, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$
12 $\sigma_{\mathrm{MSG}} \leftarrow_\$ \Sigma.\mathsf{Sign}(\mathsf{pp}_\Sigma, t, \mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma)$
13 $C \leftarrow_\$ \Pi.\mathsf{Enc}(\mathsf{pp}_\Pi, (\mathsf{pk}_{\mathsf{Sig}}, K_{\mathrm{ADM}}, \mathrm{ADM}), \mathsf{pk}_\Pi)$
14 $u \leftarrow (1, \mathsf{pk}_{\mathsf{San}}, V, C)$
15 $\sigma_{\mathrm{ADM}} \leftarrow_\$ \Sigma.\mathsf{Sign}(\mathsf{pp}_\Sigma, u, \mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma)$
16 $\sigma \leftarrow (S, \sigma_{\mathrm{MSG}}, V, C, \sigma_{\mathrm{ADM}})$
17 return σ

Fig. 2. Invisible sanitizable signature scheme from a public-key encryption scheme: parameter generation, signer and sanitizer key generation, and signing algorithms.

$\mathsf{SSS.Verify}(\mathsf{pp}_{\mathsf{SSS}}, m, \sigma, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$:

1 parse $\mathsf{pp}_{\mathsf{SSS}} = (\mathsf{pp}_{\Pi}, \mathsf{pp}_{\Sigma}, \mathcal{M})$,
$m = (m[1], \ldots, m[l])$,
$\sigma = (S, \sigma_{\mathrm{MSG}}, V, C, \sigma_{\mathrm{ADM}})$,
$S = (\sigma^1, \ldots, \sigma^l)$,
$V = (\mathsf{pk}_{\Sigma}^1, \ldots, \mathsf{pk}_{\Sigma}^l)$,
$\mathsf{pk}_{\mathsf{Sig}} = \mathsf{pk}_{\Sigma}$,
$\mathsf{pk}_{\mathsf{San}} = (\mathsf{pk}_{\Pi}, \mathsf{pk}'_{\Sigma})$
2 $t \leftarrow (0, m, S, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$
3 $d_1 \leftarrow \Sigma.\mathsf{Verify}(\mathsf{pp}_{\Sigma}, t, \sigma_{\mathrm{MSG}}, \mathsf{pk}_{\Sigma})$
4 $d_2 \leftarrow \Sigma.\mathsf{Verify}(\mathsf{pp}_{\Sigma}, t, \sigma_{\mathrm{MSG}}, \mathsf{pk}'_{\Sigma})$
5 $u \leftarrow (1, \mathsf{pk}_{\mathsf{San}}, V, C)$
6 $d_3 \leftarrow \Sigma.\mathsf{Verify}(\mathsf{pp}_{\Sigma}, u, \sigma_{\mathrm{ADM}}, \mathsf{pk}_{\Sigma})$
7 if $(d_1 = \bot \ \wedge \ d_2 = \bot) \vee$
$d_3 = \bot \ \vee \ \Sigma.\mathsf{Verify}(\mathsf{pp}_{\Sigma}, m[i],$
$\sigma^i, \mathsf{pk}_{\Sigma}^i) = \bot$ for some $1 \le i \le l$
then
8 return $\bot$
9 return $\top$

$\mathsf{SSS.Judge}(\mathsf{pp}_{\mathsf{SSS}}, m, \sigma, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}}, \pi)$:

1 parse $\mathsf{pp}_{\mathsf{SSS}} = (\mathsf{pp}_{\Pi}, \mathsf{pp}_{\Sigma}, \mathcal{M})$,
$\sigma = (S, \sigma_{\mathrm{MSG}}, V, C, \sigma_{\mathrm{ADM}})$,
$\mathsf{pk}_{\mathsf{Sig}} = \mathsf{pk}_{\Sigma}$,
$\mathsf{pk}_{\mathsf{San}} = (\mathsf{pk}_{\Pi}, \mathsf{pk}'_{\Sigma})$
2 $t \leftarrow (0, m, S, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$
3 $d_1 \leftarrow \Sigma.\mathsf{Verify}(\mathsf{pp}_{\Sigma}, t, \sigma_{\mathrm{MSG}}, \mathsf{pk}_{\Sigma})$
4 $d_2 \leftarrow \Sigma.\mathsf{Verify}(\mathsf{pp}_{\Sigma}, t, \sigma_{\mathrm{MSG}}, \mathsf{pk}'_{\Sigma})$
5 if $d_1 = \top \ \wedge \ d_2 = \bot$ then
6 return Sig
7 if $d_1 = \bot \ \wedge \ d_2 = \top$ then
8 return San
9 return $\bot$

$\mathsf{SSS.Sanit}(\mathsf{pp}_{\mathsf{SSS}}, m, \sigma, \mathsf{sk}_{\mathsf{San}}, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}}, \mathrm{MOD})$:

1 if $\mathrm{MOD}(m) = \bot$ then return $\bot$
2 parse $\mathsf{pp}_{\mathsf{SSS}} = (\mathsf{pp}_{\Pi}, \mathsf{pp}_{\Sigma}, \mathcal{M})$,
$m = (m[1], \ldots, m[l])$,
$\sigma = (S, \sigma_{\mathrm{MSG}}, V, C, \sigma_{\mathrm{ADM}})$,
$V = (\mathsf{pk}_{\Sigma}^1, \ldots, \mathsf{pk}_{\Sigma}^l)$,
$\mathsf{sk}_{\mathsf{San}} = (\mathsf{sk}_{\Pi}, \mathsf{sk}'_{\Sigma})$, $\mathsf{pk}_{\mathsf{Sig}} = \mathsf{pk}_{\Sigma}$,
$\mathsf{pk}_{\mathsf{San}} = (\mathsf{pk}_{\Pi}, \mathsf{pk}'_{\Sigma})$,
$\mathrm{MOD} = (M, l)$,
$M = \{(i_1, m_1), \ldots, (i_k, m_k)\}$
3 $t \leftarrow \Pi.\mathsf{Dec}(\mathsf{pp}_{\Pi}, C, \mathsf{pk}_{\Pi}, \mathsf{sk}_{\Pi})$
4 parse $t = (\mathsf{pk}'_{\mathsf{Sig}}, K_{\mathrm{ADM}}, \mathrm{ADM})$,
$K_{\mathrm{ADM}} = (K_{\mathrm{ADM}}^1, \ldots, K_{\mathrm{ADM}}^l)$,
with $K_{\mathrm{ADM}}^i = \mathsf{sk}_{\Sigma}^i$ for $i \in \mathrm{ADM}$
5 $d_1 \leftarrow \mathsf{SSS.Verify}(\mathsf{pp}_{\mathsf{SSS}}, m, \sigma, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$
6 if $d_1 = \bot \ \vee \ \mathrm{MOD}(\mathrm{ADM}) = \bot \vee$
$\mathsf{pk}'_{\mathsf{Sig}} \neq \mathsf{pk}_{\mathsf{Sig}}$ then
7 return $\bot$
8 $m' \leftarrow \mathrm{MOD}(m)$
9 for $1 \le j \le k$ do
10 $\sigma^{i_j} \leftarrow_{\$} \Sigma.\mathsf{Sign}(\mathsf{pp}_{\Sigma}, m_j, \mathsf{sk}_{\Sigma}^{i_j}, \mathsf{pk}_{\Sigma}^{i_j})$
11 $S' \leftarrow (\sigma^1, \ldots, \sigma^l)$
12 $u \leftarrow (0, m', S', \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$
13 $\sigma'_{\mathrm{MSG}} \leftarrow_{\$} \Sigma.\mathsf{Sign}(\mathsf{pp}_{\Sigma}, u, \mathsf{sk}'_{\Sigma}, \mathsf{pk}'_{\Sigma})$
14 $\sigma' \leftarrow (S', \sigma'_{\mathrm{MSG}}, V, C, \sigma_{\mathrm{ADM}})$
15 $d_2 \leftarrow \mathsf{SSS.Verify}(\mathsf{pp}_{\mathsf{SSS}}, m', \sigma', \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$
16 if $d_2 = \bot$ then
17 return $\bot$
18 return σ'

$\mathsf{SSS.Proof}(\mathsf{pp}_{\mathsf{SSS}}, m, \sigma, \{(m_i, \sigma_i)\}_{i=1}^k, \mathsf{sk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{San}})$:

1 return $\bot$

Fig. 3. Invisible sanitizable signature scheme from a public-key encryption scheme: verification, sanitization, judge and proof algorithms.

Construction 4. *Let* $\Sigma := (\mathsf{PGen}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a signature scheme and* $\Pi := (\mathsf{PGen}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *a public-key encryption scheme. We define a sanitizable signature scheme* SSS *as in Figs. 2 and 3 above.*

4.2 Security

The formal security statement for our construction is given in Theorem 5. Its proof can be found in the full version [21].

Theorem 5. *If the signature scheme Σ is correct and unforgeable, and the encryption scheme Π is correct, then the sanitizable signature scheme* SSS *in Construction 4 is correct. If Σ is unforgeable and Π is* IND−CPA*-secure, then* SSS *is unforgeable, immutable, private, publicly accountable, and invisible.*

4.3 Achieving Strong Invisibility

In the previous sections we have shown that invisibility is equivalent to IND−CPA-secure encryption, and that strong invisibility implies IND−CCA2-secure encryption. Here we show that the latter implication also holds in the other direction: If we use an IND−CCA2-secure encryption scheme in our construction, then we get a strongly invisible sanitizable signature scheme.

Theorem 6. *If the signature scheme Σ is correct and unforgeable, and the encryption scheme Π is correct, then the sanitizable signature scheme* SSS *in Construction 4 is correct. If Σ is unforgeable and Π is* IND−CCA2*-secure, then* SSS *is unforgeable, immutable, private, publicly accountable, and strongly invisible.*

The proof of Theorem 6 can be found in the full version [21].

5 Conclusions

Our results show that building invisible sanitizable signature schemes from one-way functions alone is presumably hard, since deriving public-key encryption from one-wayness in a black-box way is infeasible [27]. This is in contrast to sanitizable schemes without the invisibility and transparency properties. Namely, Brzuska *et al.* [6] gave a simple construction of a non-invisible, non-transparent scheme based on regular signature schemes only.

An interesting open question concerns the minimal assumptions required to achieve *transparency* for sanitizable signatures, independently of the question regarding invisibility. It is possible to achieve all the common security properties, except for transparency (and except for invisibility, of course), using one-way functions alone [6,9]. Current constructions achieving transparency are based on assumptions seemingly stronger than one-way functions, such as group signature schemes [7], zero-knowledge proofs [22], or (chameleon) hash functions [3,10]. Finally, for a sanitizable signature scheme to be both transparent and invisible, public-key encryption is at least necessary, as discussed here.

Acknowledgments. We thank the anonymous reviewers for their valuable comments and suggestions. This work has been co-funded by the DFG as part of project P2 within the CRC 1119 CROSSING.

References

1. Ahn, J.H., Boneh, D., Camenisch, J., Hohenberger, S., Shelat, A., Waters, B.: Computing on authenticated data. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 1–20. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_1
2. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: De Capitani di Vimercati, S., Syverson, P., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005). https://doi.org/10.1007/11555827_10
3. Beck, M.T., Camenisch, J., Derler, D., Krenn, S., Pöhls, H.C., Samelin, K., Slamanig, D.: Practical strongly invisible and strongly accountable sanitizable signatures. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10342, pp. 437–452. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60055-0_23
4. 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). https://doi.org/10.1007/978-3-642-13708-2_6
5. 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). https://doi.org/10.1007/978-3-642-00468-1_18
6. Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Santizable signatures: how to partially delegate control for authenticated data. In: BIOSIG 2009, pp. 117–128 (2009)
7. 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). https://doi.org/10.1007/978-3-642-13013-7_26
8. 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). https://doi.org/10.1007/978-3-642-40012-4_12
9. Brzuska, C., Pöhls, H.C., Samelin, K.: Efficient and perfectly unlinkable sanitizable signatures without group signatures. In: Katsikas, S., Agudo, I. (eds.) EuroPKI 2013. LNCS, vol. 8341, pp. 12–30. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-53997-8_2
10. Camenisch, J., Derler, D., Krenn, S., Pöhls, H.C., Samelin, K., Slamanig, D.: Chameleon-hashes with ephemeral trapdoors. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10175, pp. 152–182. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54388-7_6
11. 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). https://doi.org/10.1007/978-3-642-11925-5_13
12. 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). https://doi.org/10.1007/978-3-642-31410-0_3
13. Canard, S., Laguillaumie, F., Milhau, M.: *Trapdoor* sanitizable signatures and their application to content protection. In: Bellovin, S.M., Gennaro, R., Keromytis, A., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 258–276. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68914-0_16

14. Coretti, S., Maurer, U., Tackmann, B., Venturi, D.: From single-bit to multi-bit public-key encryption via non-malleable codes. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9014, pp. 532–560. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46494-6_22
15. Damgård, I., Haagh, H., Orlandi, C.: Access control encryption: enforcing information flow with cryptography. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 547–576. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_21
16. de Meer, H., Pöhls, H.C., Posegga, J., Samelin, K.: On the relation between redactable and sanitizable signature schemes. In: Jürjens, J., Piessens, F., Bielova, N. (eds.) ESSoS 2014. LNCS, vol. 8364, pp. 113–130. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04897-0_8
17. Demirel, D., Derler, D., Hanser, C., Pöhls, H.C., Slamanig, D., Traverso, G.: Overview of functional and malleable signature schemes (PRISMACLOUD deliverable d4.4). Technical report (2015)
18. Derler, D., Pöhls, H.C., Samelin, K., Slamanig, D.: A general framework for redactable signatures and new constructions. In: Kwon, S., Yun, A. (eds.) ICISC 2015. LNCS, vol. 9558, pp. 3–19. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30840-1_1
19. Derler, D., Slamanig, D.: Rethinking privacy for extended sanitizable signatures and a black-box construction of strongly private schemes. In: Au, M.-H., Miyaji, A. (eds.) ProvSec 2015. LNCS, vol. 9451, pp. 455–474. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26059-4_25
20. Fehr, V., Fischlin, M.: Sanitizable signcryption: sanitization over encrypted data (full version). Cryptology ePrint Archive, Report 2015/765 (2015). http://eprint.iacr.org/2015/765
21. Fischlin, M., Harasser, P.: Invisible sanitizable signatures and public-key encryption are equivalent. Cryptology ePrint Archive, Report 2018/337 (2018). https://eprint.iacr.org/2018/337
22. Fleischhacker, N., Krupp, J., Malavolta, G., Schneider, J., Schröder, D., Simkin, M.: Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9614, pp. 301–330. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49384-7_12
23. Ghosh, E., Goodrich, M.T., Ohrimenko, O., Tamassia, R.: Fully-dynamic verifiable zero-knowledge order queries for network data. Cryptology ePrint Archive, Report 2015/283 (2015). http://eprint.iacr.org/2015/283
24. Ghosh, E., Ohrimenko, O., Tamassia, R.: Zero-knowledge authenticated order queries and order statistics on a list. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 2015. LNCS, vol. 9092, pp. 149–171. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28166-7_8
25. 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). https://doi.org/10.1007/978-3-642-21518-6_21
26. Hohenberger, S., Lewko, A., Waters, B.: Detecting dangerous queries: a new approach for chosen ciphertext security. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 663–681. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_39
27. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: 21st ACM STOC, pp. 44–61 (1989)

28. 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). https://doi.org/10.1007/3-540-45760-7_17
29. 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). https://doi.org/10.1007/11927587_28
30. Krenn, S., Samelin, K., Sommer, D.: Stronger security for sanitizable signatures. In: Garcia-Alfaro, J., Navarro-Arribas, G., Aldini, A., Martinelli, F., Suri, N. (eds.) DPM/QASA -2015. LNCS, vol. 9481, pp. 100–117. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29883-2_7
31. Matsuda, T., Hanaoka, G.: An asymptotically optimal method for converting bit encryption to multi-bit encryption. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 415–442. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_18
32. Myers, S., Shelat, A.: Bit encryption is complete. In: 50th FOCS, pp. 607–616 (2009)
33. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: 21st ACM STOC, pp. 33–43 (1989)
34. 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). https://doi.org/10.1007/978-3-642-21554-4_10
35. Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: 22nd ACM STOC, pp. 387–394 (1990)
36. 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). https://doi.org/10.1007/978-3-642-13708-2_4