An extended abstract of this paper appears at PKC 2016.

Efficient Unlinkable Sanitizable Signatures from Signatures with Re-Randomizable Keys

(Full Version)

Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin

CISPA, Saarland University

Abstract. In a sanitizable signature scheme the signer allows a designated third party, called the sanitizer, to modify certain parts of the message and adapt the signature accordingly. Ateniese et al. (ESORICS 2005) introduced this primitive and proposed five security properties which were formalized by Brzuska et al. (PKC 2009). Subsequently, Brzuska et al. (PKC 2010) suggested an additional security notion, called unlinkability which says that one cannot link sanitized message-signature pairs of the same document. Moreover, the authors gave a generic construction based on group signatures that have a certain structure. However, the special structure required from the group signature scheme only allows for inefficient instantiations.

Here, we present the first efficient instantiation of unlinkable sanitizable signatures. Our construction is based on a novel type of signature schemes with re-randomizable keys. Intuitively, this property allows to re-randomize both the signing and the verification key separately but consistently. This allows us to sign the message with a re-randomized key and to prove in zero-knowledge that the derived key originates from either the signer or the sanitizer. We instantiate this generic idea with Schnorr signatures and efficient Σ-protocols, which we convert into non-interactive zero-knowledge proofs via the Fiat-Shamir transformation. Our construction is at least one order of magnitude faster than instantiating the generic scheme of Brzuska et al. with the most efficient group signature schemes.

1 Introduction

Sanitizable signature schemes were introduced by Ateniese et al. [ACdT05] and similar primitives were concurrently proposed by Steinfeld, Bull, and Zheng [SBZ02], by Miyazaki et al. [MSK02], and by Johnson et al. [JMSW02]. The basic idea of this primitive is that the signer specifies parts of a (signed) message such that a dedicated third party, called the sanitizer, can change the message and adapt the signature accordingly. Sanitizable signatures have numerous applications, such as the anonymization of medical data, replacing commercials in authenticated media streams, or updates of reliable routing information [ACdT05]. After the first introduction of sanitizable signatures in [ACdT05], the desired security properties were later formalized by Brzuska et al. [BFF$^+$09]. At PKC 2010, Brzuska et al. [BFLS10] identified an important missing property called unlinkability. Loosely speaking, this notion ensures that one cannot link sanitized message-signature pairs of the same document. This property is essential in applications like the sanitization of medical records because it prevents the attacker from combining information of several sanitized versions of a document in order to reconstruct (parts of) the original document. The authors also showed that unlinkable sanitizable signatures can be constructed from group signatures [BMW03] having the property that the keys of the signers can be computed independently, and in particular before the keys of the group manager. However, to this date, no efficient group signature scheme *that has the required properties* is known, which also means that no efficient unlinkable sanitizable signature scheme is known. This leaves us in an unsatisfactory situation. Either we use efficient sanitizable signature schemes that only achieve a subset of the security properties [ACdT05, BFF$^+$09] or we have to rely on an inefficient black-box construction of unlinkable sanitizable signatures.

In this work, we close this gap by presenting the first efficient unlinkable sanitizable signature scheme that achieves all security properties. The instantiation of our scheme only requires 15 exponentiations for signing, 17 for the verification, and 14 for sanitizing a message-signature pair. This is at least one order of magnitude faster than the fastest previously known construction. For a detailed performance comparison, refer to Section 1.2.

1.1 Overview of our Construction

In this section, we describe the main idea of our construction and the underlying techniques. Our solution is based on a novel type of digital signature schemes called *signatures with perfectly re-randomizable keys*. This type of signature schemes allows to re-randomize both the signing and the verification key separately. It is required that the re-randomization is perfect, meaning that re-randomized keys must have the same distribution as the original key. The new unforgeability notion for this type of signature scheme requires that it is infeasible for an attacker to output a forgery under either the original or a re-randomized key, even if the randomness is controlled by the attacker.

We show that this notion does not trivially follow from the regular notion of unforgeability. In fact, only a few signature schemes having this property achieve our notion of unforgeability under re-randomizable keys. We demonstrate this fact by showing concrete attacks against some well known unforgeable signature schemes that have re-randomizable keys. In particular, we show that the signature scheme of Boneh and Boyen [BB04] and the one of Camenisch and Lysyanskaya [CL04] have re-randomizable keys, but are insecure with respect to our stronger security notion. We stress that these attacks have no implications on the original security proof, but that they cannot be used as an instantiation. On the positive side, we prove that Schnorr's signature scheme [Sch90,Sch91] has re-randomizable keys and fulfills our security notion. It is well known that Schnorr's signature scheme [Sch90,Sch91] is one of the most efficient signature schemes based on the discrete logarithm assumption. Moreover, we also propose an instantiation of signature schemes with re-randomizable keys in the standard model by slightly modifying the signature scheme of Hofheinz and Kiltz [HK08,HK12].

Apart from their usefulness in constructing highly efficient sanitizable signatures, this primitive may also be of independent interest. A second possible application of signature schemes with re-randomizable keys are stealth addresses [Fra15] in Bitcoin or other cryptocurrencies. On a very high level, Bitcoin replaces bank accounts with keys of a signature scheme. Money transactions in Bitcoin transfer money from one public key to another and are only valid if they are signed with the secret key of the payer. All transactions are logged in a public log data structure, the block chain, which can be used to verify the validity of new transactions as well as to track money flow in Bitcoin. Our signatures with re-randomizable keys provide a conceptually very simple solution for so called stealth addresses. Consider a Bitcoin donation address on a website to support the host of the website or donate money to the website for a good cause. A donor may be unwilling to donate money if he can be linked to the website or other donors by the block chain. Using signatures with re-randomizable keys a donor can take the donation address, re-randomize it, and pay the money to the re-randomized address and transmit the re-randomization factor to the recipient through a non-public channel, such as email. The recipient can use the given re-randomization factor to re-randomize his corresponding secret key to further transfer the received money. Such addresses that are related in some invisible way to the recipient are called stealth addresses. For a more detailed treatment of Bitcoin and the existing stealth address mechanism see [Fra15].

Construction of Unlinkable Sanitizable Signature Schemes. Our construction is based on signature schemes that have perfectly re-randomizable keys. To sign a message m, the signer first splits the message into the parts that cannot be modified by the sanitizer and those that may be changed. Subsequently, the signer authenticates the entire messages using a signature scheme with re-randomized keys. However, the signer cannot sign this part directly as this would reveal the identity of the signer. Instead, the signer chooses a randomness ρ, re-randomizes their key-pair, and then proves, in zero-knowledge, that the derived public key is a re-randomization of either the signer's or the sanitizer's key.

Sanitizing a message follows the same idea: the sanitizer modifies the message and signs it with a re-randomized version of their key pair and appends a zero-knowledge proof for the same language.

To turn this idea into an efficient scheme, we propose an efficient sigma protocol tailored to our problem that we then convert via the Fiat-Shamir transformation [FS87] into an efficient non-interactive zero-knowledge proof. The main observation is that our zero-knowledge proofs prove only simple statements about the keys and *not* about encrypted signatures that verify under either the signer or the sanitizers public-key. Since the corresponding language is much simpler than this standard “encrypt-and-proof” approach, it has much shorter statements and thus the resulting zero-knowledge proofs are significantly more efficient.

1.2 Evaluation and Comparison

To demonstrate the efficiency of our approach, we compare both the computational and the storage complexity of our construction to the one of Brzuska et al. [BFLS10], where we use the currently most efficient instantiations of the underlying (group) signature scheme. Somewhat surprisingly, only a few group signature schemes have the property that the user keys can be generated independently of and, in particular, before the group manager’s key — a property that is required by [BFLS10]. This property originates from the definitions of Bellare, Micciancio, and Warinschi [BMW03] and only very few group signature schemes, such as [Gro07, FY05], can be adapted to have this property and at the same time fulfill all security requirements needed in [BFLS10]. In most cases the group member’s keys depend on some information published by the group manager. Finally, we instantiate the signature scheme in [BFLS10] using a deterministic version of Schnorr’s signature scheme. Thus, in our comparison shown in Table 1, we instantiate [BFLS10] with the group signature schemes of Groth [Gro07] and of Furukawa and Yonezawa [FY05], which are to the best of our knowledge the two most efficient group signature schemes that can be adapted to allow an instantiation of [BFLS10]. Our comparison shows that in the most important algorithms, i.e., signing, sanitizing, and verification, our construction is at least one order of magnitude faster than both instantiations of [BFLS10]. Similarly, Table 2 provides an overview of the storage complexity of the different constructions. Although our keys are slightly larger than the other instances, it also shows that our signatures are significantly smaller than the ones of the other instances. Note that both the number of exponentiations and the number of group elements for Furukawa and Yonezawa’s group signature scheme depend linearly on the security parameter. In our comparison, the scheme is instantiated with 100 bit security.

	KGen_{sig}	KGen_{san}	Sign	Sanit	Verify	Proof	Judge
This paper	7E	1E	15E	14E	17E	23E	6E
[BFLS10] using [Gro07]	1E	1E	194E+2P	186E+1P	207E+62P	14E+1P	1E+2P
[BFLS10] using [FY05]	1E	4E	2831E	2814E	2011E	18E	2E

Table 1. Comparison of the dominant operations in our construction instantiated as described in Section 5 with the construction of Brzuska et al. [BFLS10] instantiated with Schnorr signatures and the group signature schemes of Groth [Gro07] and Furukawa and Yonezawa [FY05] respectively. E and P stand for group exponentiations and pairing evaluations respectively.

	pk_{sig}	sk_{sig}	pk_{san}	sk_{san}	σ	π
This paper	7	14	1	1	14	4
[BFLS10] using [Gro07]	1	1	1	1	69	1
[BFLS10] using [FY05]	1	1	5	1	1620	3

Table 2. Comparison of the key, signature, and proof sizes in our construction instantiated as described in Section 5 with the construction of Brzuska et al. [BFLS10] instantiated with Schnorr signatures and the group signature schemes of Groth [Gro07] and Furukawa and Yonezawa [FY05] respectively. Here pk_{sig}, sk_{sig}, pk_{san}, and sk_{san} refer to the signer’s and sanitizer’s public and secret keys, while σ refers to the signature, and π refers to the proof that is used to determine accountability. The sizes are measured in group elements. For the sake of simplicity we do not distinguish between elements of different groups such as $\mathbb{Z}_q$ and $\mathbb{G}$. This simplification slightly favors [BFLS10] using [Gro07], since group signatures in this scheme consist exclusively of $\mathbb{G}$-elements.

Thus, it is easy to see that our solutions is the first scheme that is efficient enough to be used in practice today.

1.3 Related Work

Ateniese et al. [ACdT05] first introduced sanitizable signatures and gave an informal description of the following properties: *Unforgeability* ensures that only the honest signer and sanitizer can create valid signatures. *Immutability* says that the (malicious) sanitizer can only modify designated parts of the message. *Transparency* guarantees that signatures computed by the signer and the sanitizer are indistinguishable. *Accountability* demands that, with the help of the signer, a proof of authorship can be generated, such that neither the malicious signer nor the malicious sanitizer can deny authorship of the message. These properties were later formalized by Brzuska et al. [BFF$^+$09] and the *Unlinkability* property was introduced by Brzuska et al. in [BFLS10]. Later, in [BPS12], Brzuska et al. introduce the notion of non-interactive public accountability, which allows a third party, without help from the signer, to determine, whether a message originates from the signer or the sanitizer. In [BPS13], the same authors provide a slightly stronger unlinkability notion and an instantiation that has non-interactive public accountability and achieves their new unlinkability notion. However, non-interactive accountability and transparency are mutually exclusive. That is, no scheme can fulfill both properties at the same time. In this work we focus on schemes that have (interactive) accountability and transparency. Another line of research initiated by Klonowski and Lauks [KL06] and continued by Canard and Jambert [CJ10] considers different methods for limiting the allowed operations of the sanitizer. That is, they show how to limit the set of possible modifications on one single block and how to enforce the same modifications on different message blocks. In [CJL12], Canard et al. extend sanitizable signatures to the setting with multiple signers and sanitizers. Recently, Derler and Slamanig suggested a security notion that is stronger than privacy but weaker than unlinkability [DS15].

Other closely related types of malleable signature schemes, such as homomorphic signatures [JMSW02, BF11,JWL12,ALP13,Fre12,Cat14] or redactable signatures [SBZ02,JMSW02,PS14,BBD$^+$10,CLX09], where parts of the signed message can be removed, are closely related to sanitizable signatures, but aim to solve related but different problems, have different security notions, and are not directly applicable to solve the problem of efficient unlinkable sanitizable signatures. In [BPW03] Boldyreva et al. deal with proxy signature schemes for delegating signing rights. In such signature schemes a designator can delegate signing rights to a proxy signer, who can then sign messages on behalf of the designator. However, in such a scheme the proxy signatures are publicly distinguishable from signatures created by the designator. This would break the transparency property of sanitizable signature schemes. Policy-based signatures [BF14] allows a signer to delegate signing rights in connection with a policy that specifies, which messages can be signed with the delegated signing key. In addition, they require that they delegation policy shall remain hidden. In a similar vein to [BF14] in [BGI14] the authors explore the possibilities of delegating signing keys for arbitrary functions. That is, using the delegated signing key one can sign functions of the message that correspond to the key. These works show theoretical solutions to the discussed problems, but are too slow for practical use due to the cryptographic tools they use.

2 Sanitizable Signatures

Sanitizable signature schemes allow the delegation of signing capabilities to a designated third party, called the sanitizer. These delegation capabilities are realized by letting the signer "attach" a description of the admissible modifications ADM for this particular message and sanitizer. The sanitizer may then change the message according to some modification MOD and update the signature using their private key. More formally, the signer holds a key pair $(\mathsf{sk}_{sig}, \mathsf{pk}_{sig})$ and signs a message m and the description of the admissible modifications ADM for some sanitizer pk_{san} with its private key sk_{sig}. The sanitizer, having a matching private key sk_{san}, can update the message according to some modification MOD and compute a signature using his secret key sk_{san}. In case of a dispute about the origin of a message-signature pair, the signer can compute a proof π (using an algorithm Proof) from previously signed messages that proves that a signature has been created by the sanitizer. The verification of this proof is done by an algorithm Judge (that only decides the origin of a valid message-signature pair in question; for invalid pairs such decisions are in general impossible).

Admissible Modifications. Following [BFF$^+$09,BFLS10] closely, we assume that $\textsc{Adm}$ and $\textsc{Mod}$ are (descriptions of) efficient deterministic algorithms such that $\textsc{Mod}$ maps any message m to the modified message $m' = \textsc{Mod}(m)$, and $\textsc{Adm}(\textsc{Mod}) \in \{0,1\}$ indicates if the modification is admissible and matches $\textsc{Adm}$, in which case $\textsc{Adm}(\textsc{Mod}) = 1$. By $\textsc{Fix}_{\textsc{Adm}}$ we denote an efficient deterministic algorithm that is uniquely determined by $\textsc{Adm}$ and which maps m to the immutable message part $\textsc{Fix}_{\textsc{Adm}}(m)$, e.g., for block-divided messages $\textsc{Fix}_{\textsc{Adm}}(m)$ is the concatenation of all blocks not appearing in $\textsc{Adm}$. We require that admissible modifications leave the fixed part of a message unchanged, i.e., $\textsc{Fix}_{\textsc{Adm}}(m) = \textsc{Fix}_{\textsc{Adm}}(\textsc{Mod}(m))$ for all $m \in \{0,1\}^*$ and all $\textsc{Mod}$ with $\textsc{Adm}(\textsc{Mod}) = 1$. Analogously, to avoid choices like $\textsc{Fix}_{\textsc{Adm}}$ having empty output, we also require that the fixed part must be "maximal" given $\textsc{Adm}$, i.e., $\textsc{Fix}_{\textsc{Adm}}(m') \neq \textsc{Fix}_{\textsc{Adm}}(m)$ for $m' \notin \{\textsc{Mod}(m) \mid \textsc{Mod} \text{ with } \textsc{Adm}(\textsc{Mod}) = 1\}$.

2.1 Definition of Sanitizable Signatures

The following definition of sanitizable signature schemes is taken in verbatim from [BFF$^+$09,BFLS10].

Definition 1 (Sanitizable Signature Scheme). *A sanitizable signature scheme* $\mathrm{SanS} = (\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$ *consists of seven algorithms:*

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 and the corresponding public key:*

$$(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\kappa) \qquad \textit{and} \qquad (\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{KGen}_{san}(1^\kappa).$$

Signing. *The signing algorithm takes as input a message $m \in \{0,1\}^*$, a signer secret key sk_{sig}, a sanitizer public key pk_{san}, as well as a description $\textsc{Adm}$ of the admissible modifications to m by the sanitizer and outputs a signature σ. We assume that $\textsc{Adm}$ can be recovered from any signature:*

$$\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}_{sig}, \mathsf{pk}_{san}, \textsc{Adm}).$$

Sanitizing. *The sanitizing algorithm takes as input a message $m \in \{0,1\}^*$, a description $\textsc{Mod}$ of the desired modifications to m, a signature σ, the signer's public key pk_{sig}, and a sanitizer secret key sk_{san}. It modifies the message m according to the modification instruction $\textsc{Mod}$ and outputs a new signature σ' for the modified message $m' = \textsc{Mod}(m)$ or possibly $\perp$ in case of an error:*

$$\{(m', \sigma'), \perp\} \leftarrow \mathsf{Sanit}(m, \textsc{Mod}, \sigma, \mathsf{pk}_{sig}, \mathsf{sk}_{san}).$$

Verification. *The verification algorithm takes as input a message m, a candidate signature σ, a signer public key pk_{sig}, as well as a sanitizer public key pk_{san} and outputs a bit b:*

$$b \leftarrow \mathsf{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}).$$

Proof. *The proof algorithm takes as input a signer secret key sk_{sig}, a message m, a signature σ, and a sanitizer public key pk_{san} and outputs a proof π:*

$$\pi \leftarrow \mathsf{Proof}(\mathsf{sk}_{sig}, m, \sigma, \mathsf{pk}_{san}).$$

Judge. *The judge algorithm takes as input a message m, a signature σ, signer and sanitizer public keys $\mathsf{pk}_{sig}, \mathsf{pk}_{san}$, and proof π. It outputs a decision $d \in \{\mathtt{Sign}, \mathtt{San}\}$ indicating whether the message-signature pair was created by the signer or the sanitizer:*

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

For a sanitizable signature scheme the usual correctness properties should hold, saying that genuinely signed or sanitized messages are accepted and that a genuinely created proof by the signer leads the judge to decide in favor of the signer. For a formal approach to correctness see [BFF$^+$09].

2.2 Security of Sanitizable Signatures

In this section we recall the security notions of sanitizable signatures given by Brzuska et al. [BFF$^+$09, BFLS10] and we follow their description closely. The authors defined unforgeability, privacy, immutability, accountability, transparency, and unlinkability and showed that signer and sanitizer accountability together implies unforgeability and that unlinkability implies privacy. Therefore, we only focus on the necessary definitions and omit unforgeability and privacy.

Immutability. Informally, this property says that a malicious sanitizer cannot change inadmissible blocks. This is formalized in a model where the malicious sanitizer $\mathcal{A}$ interacts with the signer to obtain signatures σ_i for messages m_i, descriptions $\textsc{Adm}_i$ and keys $\mathsf{pk}_{san,i}$ of its choice. Eventually, the attacker stops, outputting a valid pair $(\mathsf{pk}^*_{san}, m^*, \sigma^*)$ such that message m^* is not a “legitimate” transformation of one of the m_i’s under the same key $\mathsf{pk}^*_{san} = \mathsf{pk}_{san,i}$. The latter is formalized by requiring that for each query $\mathsf{pk}^*_{san} \neq \mathsf{pk}_{san,i}$ or $m^* \notin \{\textsc{Mod}(m_i) \mid \textsc{Mod} \text{ with } \textsc{Adm}_i(\textsc{Mod}) = 1\}$ for the value $\textsc{Adm}_i$ in σ_i. This requirement enforces that for block-divided messages m^* and m_i differ in at least one inadmissible block. Observe that this definition covers also the case where the adversary interact with several sanitizers simultaneously, because it can query the signer for several sanitizer keys $\mathsf{pk}_{san,i}$.

Definition 2 (Immutability). *A sanitizable signature scheme* SanS *is said to be* immutable *if for all PPT adversaries $\mathcal{A}$ the probability that the experiment $\mathsf{Immut}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$ evaluates to 1 is negligible (in κ), where*

```
Experiment $\mathsf{Immut}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$
    $(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\kappa)$
    $(\mathsf{pk}^*_{san}, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{Proof}(\mathsf{sk}_{sig},\cdot,\cdot,\cdot)}(\mathsf{pk}_{sig})$
        letting $(m_i, \textsc{Adm}_i, \mathsf{pk}_{san,i})$ and $\sigma_i$ denote the
        queries and answers to and from oracle Sign.
    Output 1 if $\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san}) = 1$ and for all i the following holds:
            $\mathsf{pk}^*_{san} \neq \mathsf{pk}_{san,i}$ or $m^* \notin \{\textsc{Mod}(m_i) \mid \textsc{Mod} \text{ with } \textsc{Adm}_i(\textsc{Mod}) = 1\}$
    Else output 0.
```

Accountability. This property demands that the origin of a (possibly sanitized) signature should be undeniable. We distinguish between *sanitizer-accountability* and *signer-accountability* and formalize each security property in the following. *Signer-accountability* says that, if a message and its signature have not been sanitized, then even a malicious signer should not be able to make the judge accuse the sanitizer.

In the sanitizer-accountability game let $\mathcal{A}_{\mathsf{Sanit}}$ be an efficient adversary playing the role of the malicious sanitizer. Adversary $\mathcal{A}_{\mathsf{Sanit}}$ has access to a Sign and Proof oracle and it succeeds if it outputs a valid message signature pair such that m^*, σ^*, together with a key pk^*_{san} (with $(\mathsf{pk}^*_{san}, m^*)$ such that the output is different from pairs $(\mathsf{pk}_{san,i}, m_i)$ previously queried to the Sign oracle). Moreover, it is required that the proof produced by the signer via Proof still leads the judge to decide “`Sign`”, i.e., that the signature has been created by the signer.

Definition 3 (Sanitizer-Accountability). *A sanitizable signature scheme* SanS *is* sanitizer-accountable *if for all PPT adversaries $\mathcal{A}$ the probability that the experiment $\mathsf{San\text{-}Acc}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$ evaluates to 1 is negligible (in κ), where*

```
Experiment $\mathsf{San\text{-}Acc}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$
    $(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\kappa)$
    $(\mathsf{pk}^*_{san}, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{Proof}(\mathsf{sk}_{sig},\cdot,\cdot,\cdot)}(\mathsf{pk}_{sig})$
        letting $(m_i, \textsc{Adm}_i, \mathsf{pk}_{san,i})$ and $\sigma_i$
        denote the queries and answers to
        and from oracle Sign
    $\pi \leftarrow \mathsf{Proof}(\mathsf{sk}_{sig}, m^*, \sigma^*, \mathsf{pk}^*_{san})$
    Output 1 if for all i the following holds:
```

```
    $(\mathsf{pk}^*_{san}, m^*) \neq (\mathsf{pk}_{san,i}, m_i)$ and
    $\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san}) = 1$ and
    $\mathsf{Judge}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san}, \pi) \neq \texttt{San}$
```

In the signer-accountability game a malicious signer $\mathcal{A}_{\mathsf{Sign}}$ gets a public sanitizing key pk_{san} as input and has access to a sanitizing oracle, which takes as input tuples $(m_i, \textsc{Mod}_i, \sigma_i, \mathsf{pk}_{sig_{,i}})$ and returns (m'_i, σ'_i). Eventually, the adversary $\mathcal{A}_{\mathsf{Sign}}$ outputs a tuple $(\mathsf{pk}^*_{sig}, m^*, \sigma^*, \pi^*)$ and is considered succesful if Judge accuses the sanitizer for the new key-message pair pk^*_{sig}, m^* with a valid signature σ^*.

Definition 4 (Signer-Accountability). *A sanitizable signature scheme* SanS *is* signer-accountable *if for all PPT adversaries $\mathcal{A}$ the probability that the experiment $\mathsf{Sig\text{-}Acc}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$ evaluates to 1 is negligible (in κ), where*

```
***Experiment*** $\mathsf{Sig\text{-}Acc}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$
  $(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{KGen}_{san}(1^n)$
  $(\mathsf{pk}^*_{sig}, m^*, \sigma^*, \pi^*) \leftarrow \mathcal{A}^{\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_{san})}(\mathsf{pk}_{san})$
      letting $(m_i, \textsc{Mod}_i, \sigma_i, \mathsf{pk}_{sig,i})$ and
      $(m'_i, \sigma'_i)$ denote the queries and
      answers to and from oracle Sanit.
  Output 1 if for all $i$ the following holds:
      $(\mathsf{pk}^*_{sig}, m^*) \neq (\mathsf{pk}_{sig,i}, m'_i)$ and
      $\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}^*_{sig}, \mathsf{pk}_{san}) = 1$ and
      $\mathsf{Judge}(m^*, \sigma^*, \mathsf{pk}^*_{sig}, \mathsf{pk}_{san}, \pi^*) \neq \texttt{Sign}$
  else output 0.
```

Transparency. Informally, this property says that one cannot decide whether a signature has been sanitized or not. Formally, this is defined in a game where an adversary $\mathcal{A}$ has access to Sign, Sanit, and Proof oracles with which the adversary can create signatures for (sanitized) messages and learn proofs. In addition, $\mathcal{A}$ gets access to a $\mathsf{Sanit/Sign}$ box which contains a secret random bit $b \in \{0,1\}$ and which, on input a message m, a modification information $\textsc{Mod}$ and a description $\textsc{Adm}$ behaves as follows:

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

Adversary $\mathcal{A}$ eventually produces an output a, the guess for b. A sanitizable signature is now *transparent* if for all efficient algorithms $\mathcal{A}$ the probability for a right guess $a = b$ in the above game is negligibly close to $\frac{1}{2}$. Below we also define a relaxed version called *proof-restricted transparency.*

Definition 5 ((Proof-Restricted) Transparency). *A sanitizable signature scheme* SanS *is said to be* proof-restrictedly transparent *if for all PPT adversaries $\mathcal{A}$ the probability that the experiment $\mathsf{Trans}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$ evaluates to 1 is negligibly bigger than 1/2 (in κ), where*

```
***Experiment*** $\mathsf{Trans}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$
 $(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\kappa)$
 $(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{KGen}_{san}(1^\kappa)$
 $b \leftarrow \{0,1\}$
 $a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_{san}),\mathsf{Proof}(\mathsf{sk}_{sig},\cdot,\cdot,\cdot),\mathsf{Sanit/Sign}(\cdot,\cdot,\cdot)}(\mathsf{pk}_{sig}, \mathsf{pk}_{san})$
  letting $M_{\mathsf{Sanit/Sign}}$ and $M_{\mathsf{Proof}}$ denote
  the sets of messages output by the Sanit/Sign
  and queried to the Proof oracle respectively.
 Output 1 if $\big(a = b$ and $M_{\mathsf{Sanit/Sign}} \cap M_{\mathsf{Proof}} = \emptyset\big)$
 Else output 0
```

Unlinkability. This security notion demands that it is not feasible to use the signatures to identify sanitized message-signature pairs originating from the same source. This should even hold if the adversary itself provides the two source message-signature pairs and modifications of which one is sanitized. It is required that the two modifications yield the same sanitized message, because otherwise predicting the source is easy, of course. This, however, is beyond the scope of signature schemes: the scheme should only prevent that *signatures* can be used to link data. In the formalization of [BFLS10], the adversary is given access to a signing oracle and a sanitizer oracle (and a proof oracle since this step depends on the signer's secret key and may leak valuable information). The adversary is also allowed to query a left-or-right oracle $\mathsf{LoRSanit}$ which is initialized with a secret random bit b and keys $\mathsf{pk}_{sig}, \mathsf{sk}_{san}$. The adversary may query this oracle on tuples $((m_0, \textsc{Mod}_0, \sigma_0), (m_1, \textsc{Mod}_1, \sigma_1))$ and returns $\mathsf{Sanit}(m_b, \textsc{Mod}_b, \sigma_b, \mathsf{pk}_{sig}, \mathsf{sk}_{san})$ if $\mathsf{Verify}(m_i, \sigma_i, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 1$ for $i = 0, 1$, $\textsc{Adm}_0 = \textsc{Adm}_1$ and if the modifications map to the same message, i.e., $\textsc{Adm}_0(\textsc{Mod}_0) = 1$, $\textsc{Adm}_1(\textsc{Mod}_1) = 1$ and $\textsc{Mod}_0(m_0) = \textsc{Mod}_1(m_1)$. Otherwise, the oracle returns $\perp$. The adversary should eventually predict the bit b significantly better than with the guessing probability of $\frac{1}{2}$.

Definition 6 (Unlinkability). *A sanitizable signature scheme* SanS *is* unlinkable *if for all PPT adversaries $\mathcal{A}$ the probability that the experiment $\mathsf{Link}_{\mathcal{A}}^{\mathrm{SanS}}(\kappa)$ evaluates to 1 is negligibly bigger than $1/2$ (in κ), where*

```
***Experiment*** $\mathsf{Link}_{\mathcal{A}}^{\mathrm{SanS}}(\kappa)$
  $(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\kappa)$
  $(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{KGen}_{san}(1^\kappa)$
  $b \leftarrow \{0, 1\}$
  $a \leftarrow \mathcal{A}^{\substack{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_{san}),\\ \mathsf{Proof}(\mathsf{sk}_{sig},\cdot,\cdot,\cdot),\mathsf{LoRSanit}(\cdot,\cdot)}}(\mathsf{pk}_{sig}, \mathsf{pk}_{san})$
  *if* $a = b$ *then output* $1$*, else output* $0$.
```

3 Signatures Schemes With Re-Randomizable Keys

In this section, we introduce signature schemes that have re-randomizable keys and which serve as the main building block for our construction. Signature schemes with this property have the advantage that one can re-randomize the key-pair $(\mathsf{sk}, \mathsf{pk})$ to a key-pair $(\mathsf{sk}', \mathsf{pk}')$ and sign a message m with a seemingly unrelated key. Jumping ahead, this property allows us to sign messages with a fresh key and prove, in zero-knowledge, the origin of the key. For one of the signature schemes we require bilinear maps, which are defined as follows. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ be an efficient, non-degenerate bilinear map, for system-wide available groups, where g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

3.1 Defining Signature Schemes With Re-randomizable Keys

To define this property and the corresponding security notion formally, we denote by $\Sigma = (\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify})$ a standard digital signature scheme, where $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\kappa), (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa), \sigma \leftarrow \mathsf{SSign}(\mathsf{sk}, m), b \leftarrow \mathsf{SVerify}(\mathsf{pk}, m, \sigma)$ are the standard algorithms of a digital signature scheme.

Definition 7 (Signatures with Perfectly Re-Randomizable Keys). *A signature scheme $\Sigma = (\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify})$ has perfectly re-randomizable keys if there exist two PPT algorithms $(\mathsf{RandSK}, \mathsf{RandPK})$ and a randomness space χ such that:*

$\mathsf{RandSK}(\mathsf{sk}, \rho)$*: The secret key re-randomization algorithm takes as input a secret key* sk *and a randomness $\rho \in \chi$ and outputs a new secret key* sk'.

$\mathsf{RandPK}(\mathsf{pk}, \rho)$*: The public key re-randomization algorithm takes as input a public key* pk *and a randomness $\rho \in \chi$ and outputs a new public key* pk'.

Correctness *The scheme is* correct *if and only if all of the following holds:*

1. *For all* $\kappa \in \mathbb{N}$*, all key-pairs* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$*, all messages* $m \in \{0,1\}^*$*, and all signatures* $\sigma \leftarrow \mathsf{SSign}(\mathsf{sk}, m)$*, it holds that* $\mathsf{SVerify}(\mathsf{pk}, m, \sigma) = 1$*.*
2. *For all* $\kappa \in \mathbb{N}$*, all key-pairs* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$*, all randomness* $\rho \in \chi$*, all messages* $m \in \{0,1\}^*$*, and* $\sigma \leftarrow \mathsf{SSign}(\mathsf{RandSK}(\mathsf{sk}, \rho), m)$*, it holds that* $\mathsf{SVerify}(\mathsf{RandPK}(\mathsf{pk}, \rho), m, \sigma) = 1$*.*
3. *For all key pairs* $(\mathsf{sk}, \mathsf{pk})$*, and a uniformly chosen randomness* $\rho \in \chi$*, the distribution of* $(\mathsf{sk}', \mathsf{pk}')$ *and* $(\mathsf{sk}'', \mathsf{pk}'')$ *is identical, where* $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}, \rho)$*,* $\mathsf{sk}' \leftarrow \mathsf{RandSK}(\mathsf{sk}, \rho)$*, and* $(\mathsf{sk}'', \mathsf{pk}'') \leftarrow \mathsf{SGen}(1^\kappa)$

3.2 Security of Signature Schemes With Re-randomizable Keys

The security of signature scheme with re-randomizable keys is defined analogously to the unforgeability of regular signature schemes, but allows the adversary to learn message/signature pairs under re-randomized keys. This should even hold if the randomness to re-randomize the keys is chosen by the attacker. In this definition, the adversary has access to two oracles. The first one, denoted by $\mathcal{O}_1$ is a regular signing oracle. The second one, denoted by $\mathcal{O}_2$ is an oracle that takes as input a message m and some randomness ρ. It then re-randomizes the private key according to ρ and signs the message using this key.

Definition 8 (Unforgeability under Re-randomized Keys). *A signature scheme with perfectly re-randomizable keys* $\Sigma = (\mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify}, \mathsf{RandSK}, \mathsf{RandPK})$ *is* unforgeable under re-randomized keys (UFRK) *if for all PPT adversaries* $\mathcal{A}$ *the probability that the experiment* $\mathsf{UFRK}^{\Sigma}_{\mathcal{A}}(\kappa)$ *evaluates to 1 is negligible (in* κ*), where*

```
Experiment $\mathsf{UFRK}^{\Sigma}_{\mathcal{A}}(\kappa)$ :
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$
    $Q := \emptyset$
    $(m^*, \sigma^*, \rho^*) \leftarrow \mathcal{A}^{\mathcal{O}_1(\mathsf{sk},\cdot), \mathcal{O}_2(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk})$
    Output 1 if one of the two conditions is fulfilled
    1. If $\mathsf{SVerify}(\mathsf{pk}, m^*, \sigma^*) = 1$
         and $m^* \notin Q$
    2. If $\mathsf{SVerify}(\mathsf{RandPK}(\mathsf{pk}, \rho^*), m^*, \sigma^*) = 1$
         and $m^* \notin Q$
    else output 0
```

```
$\mathcal{O}_1(\mathsf{sk}, m)$ :
    $Q := Q \cup \{m\}$
    $\sigma \leftarrow \mathsf{SSign}(\mathsf{sk}, m)$
    output $\sigma$

$\mathcal{O}_2(\mathsf{sk}, m, \rho)$ :
    $Q := Q \cup \{m\}$
    $\mathsf{sk}' \leftarrow \mathsf{RandSK}(\mathsf{sk}, \rho)$
    $\sigma \leftarrow \mathsf{SSign}(\mathsf{sk}', m)$
    output $\sigma$
```

Given this definition of unforgeability, one can easily obtain the "standard" notion of existential unforgeability by giving the adversary only access to $\mathcal{O}_1$ and only checking the first condition.

Definition 9 (Existential Unforgeability). *A signature scheme with perfectly re-randomizable keys* $\Sigma = (\mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify}, \mathsf{RandSK}, \mathsf{RandPK})$ *is said to be* existentially unforgeable under chosen message attacks (EUF) *if for all PPT adversaries* $\mathcal{A}$ *the probability that the experiment* $\mathsf{EUF}^{\Sigma}_{\mathcal{A}}(\kappa)$ *evaluates to 1 is negligible (in* κ*), where* $\mathsf{EUF}^{\Sigma}_{\mathcal{A}}(\kappa)$ *is defined as* $\mathsf{UFRK}^{\Sigma}_{\mathcal{A}}(\kappa)$*, but the adversary only gets access to* $\mathcal{O}_1$ *and wins if the first condition is fulfilled.*

For our construction, we also need signature schemes that are strongly unforgeable, meaning that it is computationally hard to compute a *new* signature σ^* on a message m, i.e., the adversary is allowed to submit m to the oracle and learn a signature σ and wins the game if σ^* is valid but different from σ.

Definition 10 (Strong Existential Unforgeability). *A signature scheme with perfectly re-randomizable keys* $\Sigma = (\mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify}, \mathsf{RandSK}, \mathsf{RandPK})$ *is* strongly existentially unforgeable under chosen message attacks ($\mathsf{s\text{-}EUF}$) *if for all PPT adversaries* $\mathcal{A}$ *the probability that the experiment* $\mathsf{s\text{-}EUF}^{\Sigma}_{\mathcal{A}}(\kappa)$ *evaluates to 1 is negligible (in* κ*), where* $\mathsf{s\text{-}EUF}^{\Sigma}_{\mathcal{A}}(\kappa)$ *is defined as* $\mathsf{UFRK}^{\Sigma}_{\mathcal{A}}(\kappa)$*, but the adversary only gets access to* $\mathcal{O}_1$ *and* $\mathcal{O}_1$ *maintains* $Q := Q \cup \{m, \sigma\}$*. The adversary wins only if the following condition is fulfilled:* $\mathsf{SVerify}(\mathsf{pk}, m^*, \sigma^*) = 1$ *and* $(m^*, \sigma^*) \notin Q$*.*

3.3 Counter Examples

In this section, we show that unforgeability under re-randomizable keys (Definition 8) does not trivially follow from regular unforgeability (Definition 9). In fact, very few standard model signatures, that have re-randomizable keys, are unforgeable under re-randomizable keys. We demonstrate this by giving concrete attacks against some well known schemes, such as the Boneh and Boyen [BB08] and Camenisch and Lysyanskaya [CL04] signature schemes. We remark that these attacks have no implications on the original security proof and that our attacks are outside of the regular unforgeability model.

Boneh-Boyen Signature Scheme The scheme of Boneh and Boyen [BB08] works in a bilinear groups setting and is existentially unforgeable under the q-SDH assumption. The scheme works as follows: The secret key consists of $x, y \in \mathbb{Z}_q^*$ and the public key consists of the corresponding $\mathbb{G}_2$ elements $u := g_2^x$ and $v := g_2^y$. To sign a message $m \in \mathbb{Z}_q^*$, the signer chooses a random $r \leftarrow \mathbb{Z}_q^*$, computes $s := g_1^{1/(x+m+yr)}$, and outputs the signature $\sigma = (r, s)$. To verify that a signature is valid, the verifier checks that $e(s, u \cdot g_2^m \cdot v^r) = e(g_1, g_2)$ holds. The keys of the scheme can be re-randomized additively, i.e., given randomness $(\rho_1, \rho_2) \in \mathbb{Z}_q^2$, secret keys are randomized as $(x', y') := (x + \rho_1, y + \rho_2)$ and public keys are randomized as $(u', v') := (u \cdot g_2^{\rho_1}, v \cdot g_2^{\rho_2})$.

Even though this scheme is existentially unforgeable under the q-SDH assumption and has perfectly re-randomizable keys, it is forgeable under re-randomized keys. The attack is as follows: The adversary $\mathcal{A}$ on input the public key (u, v) chooses a random message $m \in \mathbb{Z}_q^*$ as well as a random value $\rho_1 \in \mathbb{Z}_q^*$. It then queries $(m, (\rho_1, 0))$ to its signing oracle receiving back a signature $\sigma = (r, s)$. Then, it computes $m' := m + \rho_1$ and outputs $\sigma, m', (0, 0)$ as a forgery. It is easy to verify, that the verification equation actually holds for the output of $\mathcal{A}$:

$$\begin{aligned}
& & e(s, u \cdot g_2^{m'} \cdot v^r) &= e(g_1, g_2) \\
&\Leftrightarrow & e(s, g_2^{x+m+\rho_1+yr}) &= e(g_1, g_2) \\
&\Leftrightarrow & e(g_1^{\frac{1}{(x+\rho_1)+m+yr}}, g_2^{x+\rho_1+m+yr}) &= e(g_1, g_2) \\
&\Leftrightarrow & e(g_1, g_2)^{\frac{x+\rho_1+m+yr}{x+\rho_1+m+yr}} &= e(g_1, g_2) \\
&\Leftrightarrow & e(g_1, g_2) &= e(g_1, g_2)
\end{aligned}$$

Furthermore, the adversary is efficient and the only message queried to the signing oracle is m, and $m' \neq m$. Therefore, it follows that $\mathcal{A}$ breaks the unforgeability under re-randomizable keys with probability 1.

Camenisch-Lysyanskaya Signature Scheme The signature scheme of Camenisch and Lysyanskaya [CL04] works in a symmetric bilinear groups setting and is existentially unforgeable under the LRSW assumption. The scheme works as follows: The secret key consists of $x, y \in \mathbb{Z}_q$ and the public key consists of the corresponding group elements $X := g^x$ and $Y := g^y$. To sign a message $m \in \mathbb{Z}_q$, the signer chooses a random $a \leftarrow \mathbb{G}$, computes $b := a^y$ and $c := a^{x+mxy}$, and outputs the signature $\sigma = (a, b, c)$. To verify that a signature is valid, the verifier checks that $e(a, Y) = e(g, b)$ and $e(X, a) \cdot e(X, b)^m = e(g, c)$ hold. The keys of the scheme can be re-randomized multiplicatively[1]. I.e., given randomness $(\rho_1, \rho_2) \in \mathbb{Z}_q^2$, secret keys are randomized as $(x', y') := (x \cdot \rho_1, y \cdot \rho_2)$ and public keys are randomized as $(X', Y') := (X^{\rho_1}, Y^{\rho_2})$.

This scheme is also existentially unforgeable and has perfectly re-randomizable keys. Nevertheless it also is forgeable under re-randomized keys and the corresponding attack works as follows: The adversary $\mathcal{A}$ on input the public key (X, Y) chooses a random message $m \in \mathbb{Z}_q^*$ as well as a random value $\rho_2 \in \mathbb{Z}_q^* \setminus \{1\}$. It then queries $(m, (1, \rho_2))$ to its signing oracle receiving back a signature $\sigma = (a, b, c)$. It it finally computes $m' := m \cdot \rho_2$ and $b' := b^{(\rho_2^{-1})}$ and outputs $(a, b', c), m', (1, 1)$ as a forgery. It is easy to verify, that the

[1] The keys can also be re-randomized additively, however in that case neither a proof of security nor an attack are apparent.

verification equation actually holds for the output of $\mathcal{A}$. For the first check equation we have:

$$\begin{aligned} & e(a,Y) = e(g,b') \\ \Leftrightarrow \quad & e(a,g^y) = e(g,b^{(\rho_2^{-1})}) \\ \Leftrightarrow \quad & e(g^y,a) = e(g,a^{(y\rho_2)\cdot\rho_2^{-1}}) \\ \Leftrightarrow \quad & e(g^y,a) = e(g,a^y) \\ \Leftrightarrow \quad & e(g,a)^y = e(g,a)^y. \end{aligned}$$

For the second verification equation we have:

$$\begin{aligned} & e(X,a)\cdot e(X,b')^{m'} = e(g,c) \\ \Leftrightarrow \quad & e(g^x,a)\cdot e(g^x,b^{\rho_2^{-1}})^{m\cdot\rho_2} = e(g,a^{x+mxy\rho_2}) \\ \Leftrightarrow \quad & e(g,a)^x\cdot e(g^x,a^{y\rho_2\rho_2^{-1}})^{m\rho_2} = e(g,a)^{x+mxy\rho_2} \\ \Leftrightarrow \quad & e(g,a)^x\cdot e(g,a)^{mxy\rho_2} = e(g,a)^{x+mxy\rho_2} \\ \Leftrightarrow \quad & e(g,a)^{x+mxy\rho_2} = e(g,a)^{x+mxy\rho_2}. \end{aligned}$$

Furthermore, the adversary is efficient and the only message queried to the signing oracle is m, and $m' \neq m$, since $\rho_2 \neq 1$. Therefore, it follows that $\mathcal{A}$ wins the unforgeability game with re-randomizable keys with probability 1.

3.4 Instantiations

In this section, we show that our security notion is achievable in the random oracle and the standard model. In the random oracle model, we prove that Schnorr's signature scheme [Sch90, Sch91] is unforgeable under re-randomized keys and in the standard model we show that a slightly modified version of the signature scheme due to Hofheinz and Kiltz [HK08, HK12] satisfies our notion.

Random Oracle Model We show that Schnorr's signature scheme [Sch90, Sch91] is unforgeable under re-randomized keys. Our proof technique relies on an idea that was previously observed by Fischlin and Fleischhacker [FF13] in the context of an impossibility result. The core of this technique, that we call randomness switching technique, allows moving a signature from one public key to another one knowing only the difference between the two corresponding secret keys.

Definition 11 (Schnorr Signature Scheme). *Let $\mathbb{G}$ be a cyclic group of prime order q with generator g and let $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function. The Schnorr signature scheme* SSS*, working over $\mathbb{G}$, is defined as follows:*

$\mathsf{SGen}(1^\kappa)$*: Pick* $\mathsf{sk} \leftarrow \mathbb{Z}_q$ *at random, compute* $\mathsf{pk} := g^{\mathsf{sk}}$*, and output* $(\mathsf{sk},\mathsf{pk})$*.*

$\mathsf{SSign}(\mathsf{sk},m)$*: Pick $r \leftarrow \mathbb{Z}_q$ at random and compute $R := g^r$, compute $c := \mathcal{H}(R,m)$ and* $y := r + \mathsf{sk}\cdot c \mod q$*. Output $\sigma := (c,y)$.*

$\mathsf{SVerify}(\mathsf{pk},m,\sigma)$*: Parse σ as (c,y). If* $c = \mathcal{H}(\mathsf{pk}^{-c}g^y,m)$*, then output 1, otherwise output 0.*

$\mathsf{RandSK}(\mathsf{sk},\rho)$*: Compute* $\mathsf{sk}' := \mathsf{sk} + \rho \mod q$ *and output* sk'*.*

$\mathsf{RandPK}(\mathsf{pk},\rho)$*: Compute* $\mathsf{pk}' := \mathsf{pk}\cdot g^\rho$ *and output* pk'*.*

Obviously all three correctness conditions hold. It remains to show that SSS is unforgeable under re-randomized keys.

Theorem 1 (Unforgeability of Schnorr Signatures Under Re-Randomized Keys). *The signature scheme* SSS *(Definition 11) is unforgeable under re-randomized keys (Definition 8) in the random oracle model if the discrete logarithm problem in $\mathbb{G}$ is hard.*

Proof. Assume towards contradiction that there exists an efficient adversary $\mathcal{A}$ against the unforgeability under re-randomized keys. Then, we construct an adversary $\mathcal{B}$ against the existential unforgeability of SSS, which runs $\mathcal{A}$ as a black-box and simulates both oracles with its own signing oracle. More precisely, $\mathcal{B}$ answers all queries to $\mathcal{O}_1(\mathsf{sk}, m)$ with its own signing oracle and it simulates $\mathcal{O}_2(\mathsf{sk}, \rho, m)$ by first querying its own signing oracle on m, obtaining a signature (c, y), and then adapting the signatures by adding the value $\rho \cdot c$ to y. Eventually, the adversary $\mathcal{A}$ outputs a forgery (σ^*, m^*, ρ^*) with $\sigma^* = (c, y)$. The reduction $\mathcal{B}$ adapts the signature in order to serve as a forgery under the key pk by subtracting $\rho^* \cdot c$ from y. A formal description of the adversary and the simulation of the oracle $\mathcal{O}_2(\mathsf{sk}, \rho, m)$ is given in the following:

```
$\underline{\mathcal{B}^{\mathcal{O}_1(\mathsf{sk},\cdot)}(\mathsf{pk}):}$
    $(\sigma^*, m^*, \rho^*) \leftarrow \mathcal{A}^{\mathcal{O}_1(\mathsf{sk},\cdot),\mathcal{O}_2(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk})$
    Parse $\sigma^*$ as $(c, y)$
    $y' := y - \rho^* c$
    output $(c, y'), m^*$
```

```
$\underline{\mathcal{O}_2(\mathsf{sk}, \rho, m):}$
    $(c, y) \leftarrow \mathcal{O}_1(\mathsf{sk}, m)$
    $y' := y + \rho c$
    output $(c, y')$
```

For the analysis, let us assume that $\mathcal{A}$'s success probability in the experiment $\mathsf{UFRK}_{\mathcal{A}}^{\mathsf{SSS}}$ is greater than $1/\mathsf{poly}(\kappa)$. It is easy to see that $\mathcal{B}$ is efficient and that the simulation of $\mathcal{A}$'s signing oracle $\mathcal{O}_1$ is perfect. Now, we show that $\mathcal{B}$ also provides a perfect simulation of the oracle $\mathcal{O}_2$. The signature under pk received by $\mathcal{O}_2$ consists of c and y. The c value is independent of the signing key, therefore only the y value needs to be adapted. The adapted value is computed as

$$y' = y + \rho c = r + \mathsf{sk} \cdot c + \rho c = r + (\mathsf{sk} + \rho) \cdot c.$$

Obviously (c, y') is therefore a signature on m under $\mathsf{pk} \cdot g^{\rho}$ with the same randomness as (c, y). It follows that the answers to signing queries are distributed exactly as in the $\mathsf{UFRK}_{\mathcal{A}}^{\mathsf{SSS}}(\kappa)$ experiment.

Similarly the output of $\mathcal{B}$ is computed from the output of $\mathcal{A}$. Whenever $\mathcal{A}$ outputs a valid signature, message, randomness triple (σ^*, m^*, ρ^*), we have that $\sigma^* = (c, y)$ where $c = \mathcal{H}(g^r, m)$ and $y = r + (\mathsf{sk} + \rho^*) \cdot c$ for some $r \in \mathbb{Z}_q$. We therefore have

$$y' := y - \rho^* c = r + (\mathsf{sk} + \rho^*) \cdot c - \rho^* c = r + \mathsf{sk} \cdot c$$

and thus (c, y') is a valid signature on m under pk. Further, in answering signing queries for $\mathcal{A}$, the adversary $\mathcal{B}$ queries the exact same messages as $\mathcal{A}$ and therefore whenever $\mathcal{A}$ wins in the $\mathsf{UFRK}_{\mathcal{A}}^{\mathsf{SSS}}(\kappa)$ experiment, $\mathcal{B}$ wins in the $\mathsf{EUF}_{\mathcal{A}}^{\mathsf{SSS}}(\kappa)$ experiment. Combining this with the well known proof of existential unforgeability of Schnorr signatures by Pointcheval and Stern [PS96, PS00] rules out the existence of $\mathcal{A}$ under the discrete logarithm assumption in the random oracle model.

Standard Model In the following we show that a modified version of the signature schemes due to Hofheinz and Kiltz [HK08, HK12] is unforgeable under re-randomized keys. The original construction of Hofheinz and Kiltz works on type 1 and type 2 pairings and the element s in their scheme is a random bit string. However, in our case we choose s as a random element from $\mathbb{Z}_q$. This modification slightly increases the signature's size, but does not influence the original functionality or security proof. To prove the security formally, we adapt the randomness switching technique to this setting, which allows us to reduce the unforgeability under re-randomized keys to standard existential unforgeability. The scheme of Hofheinz and Kiltz requires a programmable hash function [HK08, HK12], but since security properties of programmable hash functions are not relevant to our proofs, we omit them here and refer the interested reader to [HK08, HK12].

Definition 12 (Programmable Hash Function [HK08, HK12]). *A programmable hash function* $(\mathsf{Gen}, \mathsf{Eval})$ *consists of two algorithms:*

$k \leftarrow \mathsf{Gen}(1^\kappa)$*: The key generation algorithm takes as input the security parameter* 1^κ *and generates a public key* k*.*

$y \leftarrow \mathsf{Eval}(k, m)$: *The deterministic evaluation algorithm takes as input a key k and a message $m \in \{0,1\}^\ell$ and outputs a hash value y.*

Given the definition of programmable hash functions, we define the slightly modified signature scheme due to Hofheinz Kiltz and define the re-randomization algorithms.

Definition 13 (Hofheinz Kiltz Signature Scheme [HK08, HK12]). *Let $\mathsf{PHF} = (\mathsf{Gen}, \mathsf{Eval})$ be a programmable hash function with domain $\{0,1\}^*$ and range $\mathbb{G}_1$. The signature scheme HKSS is defined as follows:*

$\mathsf{SSetup}(1^\kappa)$: *Generate a key for PHF as $k \leftarrow \mathsf{Gen}(1^\kappa)$ and output $\mathsf{pp} = k$.*

$\mathsf{SGen}(1^\kappa)$: *Pick $\mathsf{sk} \leftarrow \mathbb{Z}_q$ at random, compute $\mathsf{pk} := g_2^{\mathsf{sk}}$, and output $(\mathsf{sk}, \mathsf{pk})$.*

$\mathsf{SSign}(\mathsf{sk}, m)$: *Parse k from pp. Pick $s \leftarrow \mathbb{Z}_q$ uniformly at random and compute $y := \mathsf{Eval}(k, m)^{\frac{1}{\mathsf{sk}+s}}$. Output $\sigma := (s, y)$.*

$\mathsf{SVerify}(\mathsf{pk}, m, \sigma)$: *Parse σ as (s, y). If $e(y, \mathsf{pk} \cdot g_2^s) = e(\mathsf{Eval}(k, m), g_2)$ then output 1, otherwise output 0.*

$\mathsf{RandSK}(\mathsf{sk}, \rho)$: *Compute $\mathsf{sk}' := \mathsf{sk} + \rho \mod q$ and output sk'.*

$\mathsf{RandPK}(\mathsf{pk}, \rho)$: *Compute $\mathsf{pk}' := \mathsf{pk} \cdot g_2^\rho$ and output pk'.*

Obviously all three correctness conditions hold. It remains to show that HKSS is unforgeable under re-randomized keys.

Theorem 2 (Unforgeability of HKSS Under re-randomized Keys). *The signature scheme HKSS as defined in Definition 13 is unforgeable under re-randomized keys (Definition 8) in the standard model, if HKSS is unforgeable under chosen message attacks (Definition 9).*

Proof. Assume towards contradiction that there exists an efficient adversary $\mathcal{A}$ against the unforgeability under re-randomizable keys. Then, we construct an adversary $\mathcal{B}$ against the existential unforgeability of the underlying signature scheme, which runs $\mathcal{A}$ as a black-box. The algorithm $\mathcal{B}$ simulates the oracle $\mathcal{O}_1$ by simply forwarding the query to its own signing oracle and it uses the randomness switching technique for the simulation of $\mathcal{O}_2$. That is, whenever $\mathcal{A}$ sends a message-randomness pair (m, ρ) to $\mathcal{O}_2$, then $\mathcal{A}$ queries its signing oracle on m and adjusts the key by subtracting ρ from s. The formal description of $\mathcal{B}$ and the oracle $\mathcal{O}_2$ is given in the following:

```
$\underline{\mathcal{B}^{\mathcal{O}(\mathsf{sk},\cdot)}(\mathsf{pk}):}$
  $(\sigma^*, m^*, \rho^*) \leftarrow \mathcal{A}^{\mathcal{O}_1(\mathsf{sk},\cdot),\mathcal{O}_2(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk})$
  Parse $\sigma^*$ as $(s, y)$
  $s' := s + \rho^*$
  output $(s', y), m^*$
```

```
$\underline{\mathcal{O}_2(\mathsf{sk}, \rho, m):}$
  $(s, y) \leftarrow \mathcal{O}(m)$
  $s' := s - \rho$
  output $(s', y)$
```

For the analysis, let us assume that $\mathcal{A}$'s success probability in the experiment $\mathsf{UFRK}_{\mathcal{A}}^{\mathsf{HKSS}}(\kappa)$ is bigger than $1/\mathsf{poly}(\kappa)$. It is easy to see that $\mathcal{B}$ is efficient and that the simulation of $\mathcal{A}$'s signing oracle $\mathcal{O}_1$ is perfect. Now, we show that $\mathcal{B}$ also provides a perfect simulation of the oracle $\mathcal{O}_2$. Whenever $\mathcal{A}$ sends (ρ, m) to $\mathcal{O}_2$, then $\mathcal{B}$ returns a signature (s', y) for which it holds that $e(y, \mathsf{pk} \cdot g_2^\rho \cdot g_2^{s'}) = e(\mathsf{Eval}(k, m)^{\frac{1}{\mathsf{sk}+s}}, g_2^{\mathsf{sk}+\rho+(s-\rho)}) = e(\mathsf{Eval}(k, m), g_2)$, which has obviously the correct distribution.

Finally, we argue that $\mathcal{B}$ outputs a valid signature whenever $\mathcal{A}$ outputs a valid forgery. To see this, note that $(s' = s + \rho^*, y)$ for m^* under pk, whenever $\mathcal{A}$ returns a valid signature (s, y) for m^* under the re-randomized key $\mathsf{pk} \cdot g_2^\rho$, since $e(y, (\mathsf{pk} \cdot g_2^\rho) \cdot g_2^s) = e(y, \mathsf{pk} \cdot g_2^{\rho+s}) = e(y, \mathsf{pk} \cdot g_2^{s'})$. Combining this with the proof of existential unforgeability of the modified version of the Hofheinz Kiltz signature schemes from [HK08, HK12] rules out the existence of $\mathcal{A}$.

4 Efficient Sanitizable Signatures

In this section we show how to build efficient unlinkable sanitizable signatures from signatures with perfectly re-randomizable keys.

4.1 Preliminaries

We recall the definitions and security notions of the other building blocks required for our construction of sanitizable signatures. Namely we recall the definitions of CCA secure public key-encryption and non-interactive zero-knowledge proof systems.

CCA Secure Public-key Encryption We shortly recall the definitions of a public key encryption scheme as well as the standard notion of CCA security.

Definition 14 (Public Key Encryption Scheme). *A public key encryption scheme* $\mathcal{E} = (\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec})$ *consists of three efficient algorithms:*

$\mathsf{EGen}(1^\kappa)$*: The key generation algorithm takes as input the security parameter* 1^κ *and generates a key pair* $(\mathsf{dk}, \mathsf{ek})$*.*

$\mathsf{Enc}(\mathsf{ek}, m)$*: The encryption algorithm takes as input an encryption key* ek *and a message* $m \in \{0,1\}^*$ *and outputs a ciphertext c.*

$\mathsf{Dec}(\mathsf{dk}, c)$*: The decryption algorithm takes as input a decryption key* dk*, a ciphertext c and outputs a message* m*.*

Correctness *The scheme is* correct *if and only if for all* $\kappa \in \mathbb{N}$*, all* $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{EGen}(1^\kappa)$*, all* $m \in \{0,1\}^*$*, and all* $c \leftarrow \mathsf{Enc}(\mathsf{ek}, m)$*, it holds that* $m = \mathsf{Dec}(\mathsf{dk}, c) = 1$*.*

Definition 15 (Indistinguishability under Chosen Ciphertext Attacks). *A public key encryption scheme* $\mathcal{E} = (\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec})$ *has* indistinguishable encryptions under chosen ciphertext attacks ($\mathsf{IND\text{-}CCA}$) *if for all (possibly stateful) PPT adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *the probability that the experiment* $\mathsf{IND\text{-}CCA}^{\mathcal{E}}_{\mathcal{A}}(\kappa)$ *evaluates to 1 is negligibly bigger than* $1/2$ *(in* κ*), where*

```
Experiment $\mathsf{IND\text{-}CCA}^{\mathcal{E}}_{\mathcal{A}}(\kappa)$ :
    $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{EGen}(1^\kappa)$
    $b \leftarrow \{0,1\}$
    $m_0, m_1 \leftarrow \mathcal{A}_0^{\mathsf{Dec}(\mathsf{dk},\cdot)}(\mathsf{ek})$
    $c_b \leftarrow \mathsf{Enc}(\mathsf{ek}, m_b)$
    $a \leftarrow \mathcal{A}_1^{\mathsf{Dec}'(\mathsf{dk},c_b,\cdot)}(c_b)$
    if $a = b$, then output $1$
    else output $0$
```

```
$\mathsf{Dec}'(\mathsf{dk}, c_b, c)$ :
    if $c \neq c_b$
    then output $\mathsf{Dec}(\mathsf{dk}, c)$
    else output $\perp$
```

Non-Interactive Zero-Knowledge Proof System We recall the definitions and security properties of non-interactive zero-knowledge proof systems.

Definition 16 (Non-Interactive Zero-Knowledge Proof System). *A non-interactive zero-knowledge proof system* $(\mathsf{Setup}_{\mathsf{ZK}}, \mathsf{P}_{\mathsf{ZK}}, \mathsf{V}_{\mathsf{ZK}})$ *for a language* $\mathcal{L}$ *with the corresponding relation* $\mathcal{R}$ *consists of three algorithms:*

$\mathsf{Setup}_{\mathsf{ZK}}(1^\kappa)$*: The setup algorithm takes as input the security parameter* 1^κ *and generates a common reference string* crs*.*

$\mathsf{P}_{\mathsf{ZK}}(\mathsf{crs}, x, w)$*: The prove algorithm takes an input the common reference string* crs*, a statement* x*, and a witness* w *and outputs a zero-knowledge proof* π*.*

$\mathsf{V}_{\mathsf{ZK}}(\mathsf{crs}, x, \pi)$*: The verification algorithm takes as input the common reference string* crs*, a statement* x*, and a proof* π *and outputs* 0 *or* 1*.*

Definition 17 (Perfect Completeness). *A NIZK scheme has* perfect completeness *if and only if for all $\kappa \in \mathbb{N}$ and all adversaries $\mathcal{A}$ it holds that*

$$\Pr[\,\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^\kappa); (x, w) \leftarrow \mathcal{A}(\mathsf{crs}); \pi \leftarrow \mathsf{P}_{\mathsf{ZK}}(\mathsf{crs}, x, w); \mathsf{V}_{\mathsf{ZK}}(\mathsf{crs}, x, \pi) = 1 \mid x \in \mathcal{L}\,] = 1$$

Soundness, Zero-Knowledge and the proof of knowledge property are defined as follows:

Definition 18 (Perfect Soundness). *A NIZK scheme has* perfect soundness *if and only if for all $\kappa \in \mathbb{N}$ and all adversaries $\mathcal{A}$ it holds that*

$$\Pr[\,\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^\kappa); (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) : \mathsf{V}_{\mathsf{ZK}}(\mathsf{crs}, x, \pi) = 0 \mid x \notin \mathcal{L}\,] = 1$$

Definition 19 (Zero-knowledge). *A NIZK scheme has* computational zero-knowledge *if for all $\kappa \in \mathbb{N}$ there exists an efficient simulator $\mathsf{S} = (\mathsf{S}_0, \mathsf{S}_1)$ such that for all adversaries $\mathcal{A}$ it holds that*

$$\left| \begin{array}{l} \Pr\left[\,\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^\kappa) : \mathcal{A}^{\mathsf{P}_{\mathsf{ZK}}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1\right] \\ -\Pr\left[\,(\mathsf{crs}, \mathrm{T}) \leftarrow \mathsf{S}_0(1^\kappa) : \mathcal{A}^{\mathsf{S}'(\mathsf{crs},\mathrm{T},\cdot,\cdot)}(\mathsf{crs}) = 1\right] \end{array} \right| \leq \mathsf{negl}(\kappa),$$

where $\mathsf{S}'(\mathsf{crs}, \mathrm{T}, x, w) = \mathsf{S}_1(\mathsf{crs}, \mathrm{T}, x)$ if $(x, w) \in \mathcal{R}$ and outputs failure otherwise.

Definition 20 (Proof of Knowledge). *A NIZK scheme is a* proof of knowledge *if there exists an efficient extractor $\mathsf{Ext} = (\mathsf{Ext}_0, \mathsf{Ext}_1)$ such that the following conditions hold:*
For all polynomial time adversaries $\mathcal{A}$ it holds that

$$\left| \begin{array}{l} \Pr[\,\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^\kappa) : \mathcal{A}(\mathsf{crs}) = 1] \\ -\Pr[\,(\mathsf{crs}, \mathrm{T}) \leftarrow \mathsf{Ext}_0(1^\kappa) : \mathcal{A}(\mathsf{crs}) = 1] \end{array} \right| \leq \mathsf{negl}(\kappa).$$

For all polynomial time adversaries $\mathcal{A}$ it holds that

$$\Pr\left[\begin{array}{c} (\mathsf{crs}, \mathrm{T}) \leftarrow \mathsf{Ext}_0(1^\kappa); (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}); \\ w \leftarrow \mathsf{Ext}_1(\mathsf{crs}, \mathrm{T}, x, \pi) : (x, w) \in \mathcal{R} \end{array} \middle| \, \mathsf{V}_{\mathsf{ZK}}(\mathsf{crs}, x, \pi) = 1 \right] \geq \frac{1}{\mathsf{poly}(\kappa)}.$$

4.2 Our Construction

In the following, we describe our construction of a sanitizable signature scheme based on signatures with re-randomizable keys. Similar to previous constructions [BFF+09,BFLS10], we sign the parts of the message that cannot be changed by the sanitizer and a description of valid modifications ADM with a separate signature scheme. The main part of our construction, and which is very different from all previous schemes, is the computation of the signature on the parts that can be modified by the sanitizer. The basic idea here is that we compute this signature using a signature scheme with re-randomizable keys. That is, we compute this signature using a re-randomized private and public key-pair $(\mathsf{sk}', \mathsf{pk}')$, which was either re-randomized by the signer or the sanitizer. To allow for an easy Proof and Judge algorithm and avoid rewinding in the proof, we have to provide a way to check that pk' is in fact the re-randomization of the signer's or the sanitizer's public key. Therefore, we also include an encryption of the actual public key. In the Proof algorithm the signer can then decrypt and return this public key along with a proof of correct decryption.

In the following, for the sake of brevity all algorithms are assumed to implicitly take the public parameters as input.

Construction 1. Let $\Sigma = (\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify}, \mathsf{RandSK}, \mathsf{RandPK})$ be a signature scheme with perfectly re-randomizable keys, $\Sigma_{\mathrm{FIX}} = (\mathsf{SSetup}_{\mathrm{FIX}}, \mathsf{SGen}_{\mathrm{FIX}}, \mathsf{SSign}_{\mathrm{FIX}}, \mathsf{SVerify}_{\mathrm{FIX}})$ be a deterministic signature scheme, $\mathcal{E} = (\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme, and $\Pi_{PoK} = (\mathsf{Setup}_{\mathsf{PoK}}, \mathsf{P}_{\mathsf{PoK}}, \mathsf{V}_{\mathsf{PoK}})$ as well as $\Pi_{ZK} = (\mathsf{Setup}_{\mathsf{ZK}}, \mathsf{P}_{\mathsf{ZK}}, \mathsf{V}_{\mathsf{ZK}})$ be two non-interactive zero-knowledge proof systems for the languages $\mathcal{L}_1$ and $\mathcal{L}_2$, where the language $\mathcal{L}_1$, used in Sign, Sanit, and Verify, contains tuples $(\mathsf{ek}, c, \mathsf{pk}', \mathsf{pk}_{san}, \mathsf{pk})$ for which there exists witness $w = (\omega, \rho)$ such that

$$c = \mathsf{Enc}(\mathsf{ek}, \mathsf{pk}; \omega) \quad \wedge \quad \mathsf{pk}' = \mathsf{RandPK}(\mathsf{pk}, \rho)$$

or

$$c = \mathsf{Enc}(\mathsf{ek}, \mathsf{pk}_{san}; \omega) \quad \wedge \quad \mathsf{pk}' = \mathsf{RandPK}(\mathsf{pk}_{san}, \rho).$$

The second language $\mathcal{L}_2$, used in Proof and Judge, contains tuples $(\mathsf{ek}, c, \widehat{\mathsf{pk}})$ for which there exists witness $w = (\psi, \mathsf{dk})$ such that

$$(\mathsf{ek}, \mathsf{dk}) = \mathsf{EGen}(1^\kappa; \psi) \quad \wedge \quad \widehat{\mathsf{pk}} = \mathsf{Dec}(\mathsf{dk}, c).$$

Define our sanitizable signature scheme SanS $= (\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$ as follows:

Setup and Key Generation. The setup algorithm generates two common reference strings for the two different zero-knowledge proofs (of knowledge) and the key generation algorithm the required keys. They are formally defined as follows:

```
$\underline{\mathsf{Setup}(1^\kappa):}$
  $\mathsf{crs}_{\mathsf{PoK}} \leftarrow \mathsf{Setup}_{\mathsf{PoK}}(1^\kappa)$
  $\mathsf{crs}_{\mathsf{ZK}} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^\kappa)$
  $\mathsf{pp}_s \leftarrow \mathsf{SSetup}(1^\kappa)$
  $\mathsf{pp} = (\mathsf{crs}_{\mathsf{PoK}}, \mathsf{crs}_{\mathsf{ZK}}, \mathsf{pp}_s)$
  output $\mathsf{pp}$

$\underline{\mathsf{KGen}_{san}(1^\kappa):}$
  $(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{SGen}(1^\kappa)$
  output $(\mathsf{sk}_{san}, \mathsf{pk}_{san})$
```

```
$\underline{\mathsf{KGen}_{sig}(1^\kappa):}$
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$
  $(\mathsf{sk}_{\mathrm{FIX}}, \mathsf{pk}_{\mathrm{FIX}}) \leftarrow \mathsf{SGen}_{\mathrm{FIX}}(1^\kappa)$
  $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{EGen}(1^\kappa; \psi)$
  $\mathsf{sk}_{sig} := \begin{pmatrix} \mathsf{sk}_{\mathrm{FIX}}, \mathsf{sk}, \mathsf{dk}, \\ \mathsf{pk}_{\mathrm{FIX}}, \mathsf{pk}, \mathsf{ek}, \psi \end{pmatrix}$
  $\mathsf{pk}_{sig} := (\mathsf{pk}_{\mathrm{FIX}}, \mathsf{pk}, \mathsf{ek})$
  output $(\mathsf{sk}_{sig}, \mathsf{pk}_{sig})$
```

Signing and Sanitizing. The signing and sanitizing algorithms first parse their inputs and Sanit further checks that Mod is actually an admissible modification and modifies the message accordingly. The Sign algorithm now signs the fixed part with $\mathsf{sk}_{\mathrm{FIX}}$, while Sanit can simply reuse the σ_{FIX} of the input signature. The remainder of the two algorithms proceeds identically, by re-randomizing the respective key, encrypting the original key, proving that sk' is indeed a re-randomization and signing the full message together with signer's and sanitizer's public keys as seen in the following:

```
$\underline{\mathsf{Sign}(m, \mathsf{sk}_{sig}, \mathsf{pk}_{san}, \mathrm{ADM}):}$
  Parse $\mathsf{sk}_{sig}$ as
    $(\mathsf{sk}_{\mathrm{FIX}}, \mathsf{sk}, \mathsf{dk}, \mathsf{pk}_{\mathrm{FIX}}, \mathsf{pk}, \mathsf{ek}, \psi)$.
  $\mathsf{pk}_{sig} := (\mathsf{pk}_{\mathrm{FIX}}, \mathsf{pk}, \mathsf{ek})$
  $m_{\mathrm{FIX}} := (\mathrm{FIX}_{\mathrm{ADM}}(m), \mathrm{ADM}, \mathsf{pk}_{san})$
  $\sigma_{\mathrm{FIX}} := \mathsf{SSign}_{\mathrm{FIX}}(\mathsf{sk}_{\mathrm{FIX}}, m_{\mathrm{FIX}})$
  $\rho \leftarrow \chi$
  $\mathsf{sk}' \leftarrow \mathsf{RandSK}(\mathsf{sk}, \rho)$
  $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}, \rho)$
  $c \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{pk}; \omega)$
  $x := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}')$
  $\tau \leftarrow \mathsf{P}_{\mathsf{PoK}}(\mathsf{crs}, x, (\rho, \omega))$
  $\sigma' := \mathsf{SSign}(\mathsf{sk}', (m, \mathsf{pk}_{sig}, \mathsf{pk}_{san}))$
  output $\sigma = (\sigma_{\mathrm{FIX}}, \sigma', \mathrm{ADM}, \mathsf{pk}', c, \tau)$
```

```
$\underline{\mathsf{Sanit}(m, \mathrm{MOD}, \sigma, \mathsf{pk}_{sig}, \mathsf{sk}_{san}):}$
  Parse $\mathsf{pk}_{sig}$ as $(\mathsf{pk}_{\mathrm{FIX}}, \mathsf{pk}, \mathsf{ek})$.
  Parse $\sigma$ as $(\sigma_{\mathrm{FIX}}, \sigma', \mathrm{ADM}, \mathsf{pk}', c, \tau)$.
  If $\mathrm{ADM}(\mathrm{MOD}) = 0$
    output $\perp$
  $\widehat{m} := \mathrm{MOD}(m)$
  $\rho \leftarrow \chi$
  $\widehat{\mathsf{sk}'} \leftarrow \mathsf{RandSK}(\mathsf{sk}_{san}, \rho)$
  $\widehat{\mathsf{pk}'} \leftarrow \mathsf{RandPK}(\mathsf{pk}_{san}, \rho)$
  $\widehat{c} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{pk}_{san}; \omega)$
  $x := (\widehat{c}, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \widehat{\mathsf{pk}'})$
  $\widehat{\tau} \leftarrow \mathsf{P}_{\mathsf{PoK}}(\mathsf{crs}, x, (\rho, \omega))$
  $\widehat{\sigma}' := \mathsf{SSign}(\widehat{\mathsf{sk}}', (\widehat{m}, \mathsf{pk}_{sig}, \mathsf{pk}_{san}))$
  output $(\widehat{m}, \widehat{\sigma} = (\sigma_{\mathrm{FIX}}, \widehat{\sigma}', \mathrm{ADM}, \widehat{\mathsf{pk}'}, \widehat{c}, \widehat{\tau}))$
```

Verification. The verification algorithm checks that both signatures and the proof of knowledge verify:

```
$\underline{\mathsf{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}):}$
  Parse $\mathsf{pk}_{sig}$ as $(\mathsf{pk}_{\text{Fix}}, \mathsf{pk}, \mathsf{ek})$.
  Parse $\sigma$ as $(\sigma_{\text{Fix}}, \sigma', \text{Adm}, \mathsf{pk}', c, \tau)$.
  $m_{\text{Fix}} := (\text{Fix}_{\text{Adm}}(m), \text{Adm}, \mathsf{pk}_{san})$
  $x := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}')$
  if $\left(\begin{array}{rl} & \mathsf{SVerify}_{\text{Fix}}(\mathsf{pk}_{\text{Fix}}, m_{\text{Fix}}, \sigma_{\text{Fix}}) = 1 \\ \text{and} & \mathsf{SVerify}(\mathsf{pk}', (m, \mathsf{pk}_{sig}, \mathsf{pk}_{san}), \sigma') = 1 \\ \text{and} & \mathsf{V}_{\mathsf{PoK}}(\mathsf{crs}, x, \tau) = 1 \end{array}\right)$
  then output 1
  else output 0
```

Proving and Judging. The algorithm Proof first verifies that the given signature is indeed valid. It then parses its inputs and decrypts the ciphertext c, thus revealing who computed the signature. Moreover, it computes a zero-knowledge proof asserting that the decryption was performed correctly. The Judge checks whether the proof of decryption is correct. If the proof π contains pk_{san}, then the Judge algorithm outputs San. In all other cases, Judge returns Sign.

```
$\underline{\mathsf{Proof}(\mathsf{sk}_{sig}, m, \sigma, \mathsf{pk}_{san}):}$
  If $\mathsf{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 0$
    output $\perp$
  Parse $\mathsf{sk}_{sig}$ as
    $(\mathsf{sk}_{\text{Fix}}, \mathsf{sk}, \mathsf{dk}, \mathsf{pk}_{\text{Fix}}, \mathsf{pk}, \mathsf{ek}, \psi)$.
  Parse $\sigma$ as $(\sigma_{\text{Fix}}, \sigma', \text{Adm}, \mathsf{pk}', c, \tau)$.
  $\widehat{\mathsf{pk}} \leftarrow \mathsf{Dec}(\mathsf{dk}, c)$
  $x := (\mathsf{ek}, c, \widehat{\mathsf{pk}})$
  $\phi \leftarrow \mathsf{P}_{\mathsf{ZK}}(\mathsf{crs}, x, (\psi, \mathsf{dk}))$
  output $(\widehat{\mathsf{pk}}, \phi)$
```

```
$\underline{\mathsf{Judge}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}, \pi):}$
  Parse $\mathsf{pk}_{sig}$ as $(\mathsf{pk}_{\text{Fix}}, \mathsf{pk}, \mathsf{ek})$.
  Parse $\sigma$ as $(\sigma_{\text{Fix}}, \sigma', \text{Adm}, \mathsf{pk}', c, \tau)$.
  Parse $\pi$ as $(\widehat{\mathsf{pk}}, \phi)$.
  $x := (\mathsf{ek}, c, \widehat{\mathsf{pk}})$
  if $\left(\begin{array}{rl} & \mathsf{pk}_{san} = \widehat{\mathsf{pk}} \\ \text{and} & \mathsf{V}_{\mathsf{ZK}}(\mathsf{crs}, x, \phi) = 1 \end{array}\right)$
  then output San
  else output Sign
```

4.3 Security Proof

We now proceed by showing that our construction satisfies all necessary security definitions of a sanitizable signature scheme.

Theorem 3 (Sanitizer Accountability). *If $\Sigma = (\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify}, \mathsf{RandSK}, \mathsf{RandPK})$ is a signature scheme with perfectly re-randomizable keys that is unforgeable under re-randomized keys $\Pi_{ZK} = (\mathsf{Setup}_{\mathsf{ZK}}, \mathsf{P}_{\mathsf{ZK}}, \mathsf{V}_{\mathsf{ZK}})$ is a perfectly sound non-interactive zero knowledge proof system, and $\Pi_{PoK} = (\mathsf{Setup}_{\mathsf{PoK}}, \mathsf{P}_{\mathsf{PoK}}, \mathsf{V}_{\mathsf{PoK}})$ is a perfectly sound non-interactive zero-knowledge proof of knowledge system, then the construction given in Section 4 is sanitizer-accountable.*

Proof. Let $\mathcal{A}$ be a probabilistic polynomial time adversary against the sanitizer accountability of SanS. Let $(\mathsf{pk}^*_{san}, m^*, \sigma^*)$ denote the output of $\mathcal{A}$, where σ^* can be parsed as $(\sigma_{\text{Fix}}, \sigma', \text{Adm}, \mathsf{pk}', c, \tau)$ and let $\mathsf{pk}_{sig} = (\mathsf{pk}_{\text{Fix}}, \mathsf{pk}, \mathsf{ek})$.

By definition of Proof it holds that $\pi = (\widehat{\mathsf{pk}}, \phi)$ and $\mathsf{Dec}(\mathsf{dk}, c) = \widehat{\mathsf{pk}}$. Observe that in the case of $\mathsf{San\text{-}Acc}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa) = 1$, the following conditions must hold by definition of sanitizer accountability:

$$(\mathsf{pk}^*_{san}, m^*) \neq (\mathsf{pk}_{san,i}, m_i) \tag{1}$$
$$\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san}) = 1 \tag{2}$$
$$\mathsf{Judge}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san}, \pi) = \mathtt{Sign} \tag{3}$$

where $(m_i, \textsc{Adm}_i, \mathsf{pk}_{san,i})$ denotes the ith query to the Sign oracle.

By the definition of Verify, it follows from Equation 2 that

$$\mathsf{SVerify}(\mathsf{pk}', (m^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san}), \sigma') = 1 \tag{4}$$
$$\text{and} \quad \mathsf{V}_{\mathsf{PoK}}(\mathsf{crs}_{\mathsf{PoK}}, (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}^*_{san}, \mathsf{pk}'), \tau) = 1. \tag{5}$$

From Equation 3 it follows by the definition of Judge that at least one of the following must not hold:

$$\widehat{\mathsf{pk}} = \mathsf{pk}^*_{san} \tag{6}$$
$$\text{or} \quad \mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san}) = 1 \tag{7}$$
$$\text{or} \quad \mathsf{V}_{\mathsf{ZK}}(\mathsf{crs}_{\mathsf{ZK}}, (\mathsf{ek}, c, \widehat{\mathsf{pk}}), \phi) = 1 \tag{8}$$

However, clearly Equation 7 must hold since this is already ensured by Equation 2, and Equation 8 clearly follows from the correctness of Π_{ZK} and the fact that ϕ is computed honestly by Judge. It must thus hold that $\widehat{\mathsf{pk}} \neq \mathsf{pk}^*_{san}$. Since the correctness of $\mathcal{E}$ and the perfect soundness of Π_{PoK} guarantee, that $\widehat{\mathsf{pk}} \in \{\mathsf{pk}, \mathsf{pk}^*_{san}\}$ it therefore follows that

$$\widehat{\mathsf{pk}} = \mathsf{pk}. \tag{9}$$

Now, consider reduction $\mathcal{B}_1$, depicted in Figure 1 against the unforgeability under re-randomized keys of the underlying signature scheme. Observe that this reduction is clearly efficient and perfectly simulates the view of $\mathcal{A}$ in the game $\mathsf{San\text{-}Acc}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$. Furthermore, because of Equation 1, $(m^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san})$ is a message never queried to the signing oracle. As, further, whenever the extractor is successful in extracting the witness from τ, it follows from Equation 4 and Equation 9 that the forgery output by $\mathcal{B}_1$ is valid, it holds that

$$\Pr\left[\mathsf{UFRK}^{\Sigma}_{\mathcal{B}_1}(\kappa) = 1\right] \geq \frac{1}{\mathsf{poly}(\kappa)} \Pr\left[\mathsf{San\text{-}Acc}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa) = 1\right]$$

which must be negligible because the signature scheme is unforgeable under re-randomized keys.

Thus it must hold that $\Pr\left[\mathsf{San\text{-}Acc}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa) = 1\right]$ is negligible.

Theorem 4 (Signer Accountability). *If $\Sigma = (\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify}, \mathsf{RandSK}, \mathsf{RandPK})$ is a signature scheme with perfectly re-randomizable keys that is unforgeable under re-randomized keys and $\Pi_{PoK} = (\mathsf{Setup}_{\mathsf{PoK}}, \mathsf{P}_{\mathsf{PoK}}, \mathsf{V}_{\mathsf{PoK}})$ is a perfectly sound non-interactive zero-knowledge proof of knowledge, then the construction given in Section 4 is signer-accountable.*

Proof. Let $\mathcal{A}$ be a probabilistic polynomial time adversary against the signer accountability of SanS. Let $(\mathsf{pk}^*_{sig}, m^*, \sigma^*, \pi^*)$ denote the output of $\mathcal{A}$, where pk^*_{sig} can be parsed as $(\mathsf{pk}_{\mathrm{FIX}}{}^*, \mathsf{pk}^*, \mathsf{ek}^*)$, σ^* can be parsed as $(\sigma_{\mathrm{FIX}}, \sigma', \textsc{Adm}, \mathsf{pk}', c, \tau)$, and π^* can be parsed as $(\widehat{\mathsf{pk}}, \phi)$.

Observe that in the case of $\mathsf{Sig\text{-}Acc}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa) = 1$, the following conditions must hold by definition of signer accountability:

$$(\mathsf{pk}^*_{sig}, m^*) \neq (\mathsf{pk}_{sig,i}, m_i) \tag{10}$$
$$\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}^*_{sig}, \mathsf{pk}_{san}) = 1 \tag{11}$$
$$\mathsf{Judge}(m^*, \sigma^*, \mathsf{pk}^*_{sig}, \mathsf{pk}_{san}, \pi^*) = \mathtt{San} \tag{12}$$

```
$\mathcal{B}_1^{\mathcal{O}_1(\mathsf{sk},\cdot),\mathcal{O}_2(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk})$ :
  $\mathsf{crs}_{\mathsf{PoK}} \leftarrow \mathsf{Ext}_0(1^\kappa)$
  $(\mathsf{sk}_{\text{FIX}}, \mathsf{pk}_{\text{FIX}}) \leftarrow \mathsf{SGen}_{\text{FIX}}(1^\kappa)$
  $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{EGen}(1^\kappa; \psi)$
  $\mathsf{pk}_{sig} = (\mathsf{pk}_{\text{FIX}}, \mathsf{pk}, \mathsf{ek})$
  $(\mathsf{pk}^*_{san}, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}'(\cdot,\cdot,\cdot),\mathsf{Proof}'(\cdot,\cdot,\cdot)}(\mathsf{pk}_{sig})$
  Parse $\sigma^*$ as $(\sigma_{\text{FIX}}, \sigma', \text{ADM}, \mathsf{pk}', c, \tau)$
  $x := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}^*_{san}, \mathsf{pk}')$
  $\rho^* \leftarrow \mathsf{Ext}_1^{\mathcal{A}(\cdot)}(\mathsf{crs}, \mathrm{T}, x, \tau)$
  output $((m^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san}), \sigma', \rho^*)$
```

```
$\mathsf{Sign}'(m, \mathsf{pk}_{san}, \text{ADM})$ :
  $\rho \leftarrow \chi$
  $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}, \rho)$
  $c \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{pk}; \omega)$
  $x := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}')$
  $\tau \leftarrow \mathsf{P}_{\mathsf{PoK}}(\mathsf{crs}_{\mathsf{PoK}}, x, (\rho, \omega))$
  $m_{\text{FIX}} := (\text{FIX}_{\text{ADM}}(m), \text{ADM}, \mathsf{pk}_{san})$
  $\sigma_{\text{FIX}} \leftarrow \mathsf{SSign}_{\text{FIX}}(\mathsf{sk}_{\text{FIX}}, m_{\text{FIX}})$
  $\sigma' \leftarrow \mathcal{O}_2(\mathsf{sk}, (m, \mathsf{pk}_{sig}, \mathsf{pk}_{san}), \rho)$
  output $\sigma = (\sigma_{\text{FIX}}, \sigma', \text{ADM}, \mathsf{pk}', c, \tau)$
```

```
$\mathsf{Proof}'(m, \sigma, \mathsf{pk}_{san})$ :
  Parse $\sigma$ as $(\sigma_{\text{FIX}}, \sigma', \text{ADM}, \mathsf{pk}', c, \tau)$.
  If $\mathsf{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 0$
    return $\perp$
  $\widehat{\mathsf{pk}} \leftarrow \mathsf{Dec}(\mathsf{dk}, c)$
  $x := (\mathsf{ek}, c, \widehat{\mathsf{pk}})$
  $\phi \leftarrow \mathsf{P}_{\mathsf{ZK}}(\mathsf{crs}_{\mathsf{ZK}}, x, (\psi, \mathsf{dk}))$
  output $(\widehat{\mathsf{pk}}, \phi)$
```

Fig. 1. Description of reduction $\mathcal{B}_1$, reducing the sanitizer accountability of SanS against the UFRK security of Σ.

where $(m_i, \text{MOD}_i, \sigma_i, \mathsf{pk}_{sig,i})$ and (m'_i, σ'_i) denotes the ith query and answer to the Sanit oracle respectively.

By the definition of Verify, it follows from Equation 11 that

$$\mathsf{SVerify}(\mathsf{pk}', (m^*, \mathsf{pk}^*_{sig}, \mathsf{pk}_{san}), \sigma') = 1 \tag{13}$$

$$\mathsf{V}_{\mathsf{PoK}}(\mathsf{crs}_{\mathsf{PoK}}, (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}'), \tau) = 1. \tag{14}$$

From Equation 12 it follows by the definition of Judge that all of the following must hold:

$$\mathsf{pk}_{san} = \widehat{\mathsf{pk}} \tag{15}$$

$$\mathsf{V}_{\mathsf{ZK}}(\mathsf{crs}_{\mathsf{ZK}}, (\mathsf{ek}^*, c, \widehat{\mathsf{pk}}), \phi) = 1. \tag{16}$$

Now, consider reduction $\mathcal{B}_2$, depicted in Figure 2 against the unforgeability under re-randomized keys of the underlying signature scheme.

Observe that this reduction is clearly efficient and perfectly simulates the view of $\mathcal{A}$ in the game $\mathsf{Sig\text{-}Acc}^{\text{SanS}}_{\mathcal{A}}(\kappa)$. Furthermore, because of Equation 10, $(m^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san})$ is a message never queried to the signing oracle. As, further, whenever the extractor is successful in extracting the witness from τ, it follows from Equation 13 and Equation 15 that the forgery output by $\mathcal{B}_2$ is valid, it holds that

$$\Pr\left[\mathsf{UFRK}^{\Sigma}_{\mathcal{B}_2}(\kappa) = 1\right] \geq \frac{1}{\mathsf{poly}(\kappa)} \Pr\left[\mathsf{Sig\text{-}Acc}^{\text{SanS}}_{\mathcal{A}}(\kappa) = 1\right]$$

which must be negligible because the signature scheme is unforgeable under re-randomized keys.

Thus it must hold that $\Pr\left[\mathsf{Sig\text{-}Acc}^{\text{SanS}}_{\mathcal{A}}(\kappa) = 1\right]$ is negligible.

```
$\underline{\mathcal{B}_2^{\mathcal{O}_1(\mathsf{sk}_{san},\cdot),\mathcal{O}_2(\mathsf{sk}_{san},\cdot,\cdot)}(\mathsf{pk}_{san}) :}$
    $\mathsf{crs}_{\mathsf{PoK}} \leftarrow \mathsf{Ext}_0(1^\kappa)$
    $(\mathsf{pk}^*_{sig}, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sanit}'(\cdot,\cdot,\cdot)}(\mathsf{pk}_{san})$
    Parse $\sigma^*$ as $(\sigma_{\text{FIX}}, \sigma', \text{ADM}, \mathsf{pk}', c, \tau)$
    Parse $\mathsf{pk}^*_{sig}$ as $(\mathsf{pk}_{\text{FIX}}, \mathsf{pk}, \mathsf{ek})$
    $x := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}')$
    $\rho^* \leftarrow \mathsf{Ext}_1^{\mathcal{A}(\cdot)}(\mathsf{crs}_{\mathsf{PoK}}, \mathrm{T}_{\text{PoK}}, x, \tau)$
    output $((m^*, \mathsf{pk}^*_{sig}, \mathsf{pk}_{san}), \sigma', \rho^*)$
```

```
$\underline{\mathsf{Sanit}'(m, \sigma, \text{MOD}, \mathsf{pk}_{sig}) :}$
    Parse $\mathsf{pk}_{sig}$ as $(\mathsf{pk}_{\text{FIX}}, \mathsf{pk}, \mathsf{ek})$.
    Parse $\sigma$ as $(\sigma_{\text{FIX}}, \sigma', \text{ADM}, \mathsf{pk}', c, \tau)$.
    If $\text{ADM}(\text{MOD}) = 0$
        output $\bot$
    $\widehat{m}' := \text{MOD}(m)$
    $\rho \leftarrow \chi$
    $\widehat{\mathsf{pk}}' \leftarrow \mathsf{RandPK}(\mathsf{pk}_{san}, \rho)$
    $\widehat{c} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{pk}_{san}; \omega)$
    $x := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \widehat{\mathsf{pk}}')$
    $\widehat{\tau} \leftarrow \mathsf{P}_{\mathsf{PoK}}(\mathsf{crs}_{\mathsf{PoK}}, x, (\rho, \omega))$
    $\widehat{\sigma}' \leftarrow \mathcal{O}_2(\mathsf{sk}, (\widehat{m}', \mathsf{pk}_{sig}, \mathsf{pk}_{san}), \rho)$
    $\widehat{\sigma} = (\sigma_{\text{FIX}}, \widehat{\sigma}', \text{ADM}, \widehat{\mathsf{pk}}', \widehat{c}, \widehat{\tau})$
    output $(m', \widehat{\sigma})$
```

Fig. 2. Description of reduction $\mathcal{B}_2$, reducing the signer accountability of SanS against the UFRK security of Σ.

Theorem 5 (Immutability). *If the deterministic signature scheme* $\Sigma_{\text{FIX}} = (\mathsf{SSetup}_{\text{FIX}}, \mathsf{SGen}_{\text{FIX}}, \mathsf{SSign}_{\text{FIX}}, \mathsf{SVerify}_{\text{FIX}})$ *is strongly existentially unforgeable, then the construction given in Section 4 is immutable.*

Proof. Let $\mathcal{A}$ be a probabilistic polynomial time adversary against the immutability of SanS. Let $(\mathsf{pk}^*_{san}, m^*, \sigma^*)$ denote the output of $\mathcal{A}$, where σ^* can be parsed as $(\sigma_{\text{FIX}}, \sigma', \text{ADM}, \mathsf{pk}', c, \tau)$.

Observe that in the case of $\mathsf{Immut}_{\mathcal{A}}^{\text{SanS}}(\kappa) = 1$, it must hold by definition of immutability that

$$\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san}) = 1 \tag{17}$$

as well as at least one of the following

$$\mathsf{pk}^*_{san} \neq \mathsf{pk}_{san,i} \tag{18}$$

$$\text{or} \quad m^* \notin \{\text{MOD}(m_i) \mid \text{MOD with } \text{ADM}_i(\text{MOD}) = 1\} \tag{19}$$

where $(m_i, \text{MOD}_i, \sigma_i, \mathsf{pk}_{sig,i})$ and (m'_i, σ'_i) denotes the ith query and answer to the Sanit oracle respectively.

By the definition of Verify, it follows from Equation 17 that

$$\mathsf{SVerify}_{\text{FIX}}(\mathsf{pk}_{\text{FIX}}, (\text{FIX}_{\text{ADM}}(m^*), \text{ADM}, \mathsf{pk}^*_{san}), \sigma_{\text{FIX}}) = 1. \tag{20}$$

From Equation 19 it follows due to the maximality of FIX, that

$$\text{FIX}_{\text{ADM}}(m^*) \neq \text{FIX}_{\text{ADM}_i}(m_i) \tag{21}$$

and combining Equation 18 with Equation 21 we get that

$$(\text{FIX}_{\text{ADM}}(m^*), \text{ADM}, \mathsf{pk}^*_{san}) \neq (\text{FIX}_{\text{ADM}_i}(m_i), \text{ADM}_i, \mathsf{pk}_{san,i}) \tag{22}$$

for all i.

Now, consider reduction $\mathcal{B}_3$, depicted in Figure 3 against the strong existential unforgeability of the underlying signature scheme.

```
$\mathcal{B}_3^{\mathcal{O}(\mathsf{sk},\cdot)}(\mathsf{pk}_{\text{FIX}})$ :
    $\mathsf{crs}_{\mathsf{PoK}} \leftarrow \mathsf{Setup}_{\mathsf{PoK}}(1^\kappa)$
    $\mathsf{crs}_{\mathsf{ZK}} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^\kappa)$
    $(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$
    $(\mathsf{dk},\mathsf{ek}) \leftarrow \mathsf{EGen}(1^\kappa;\psi)$
    $\mathsf{pk}_{sig} := (\mathsf{pk}_{\text{FIX}},\mathsf{pk},\mathsf{ek})$
    $(m^*,\sigma^*,\mathsf{pk}^*_{san}) \leftarrow \mathcal{A}^{\mathsf{Sign}'(\cdot,\cdot,\cdot),\mathsf{Proof}(\mathsf{sk}_{sig},\cdot,\cdot)}(\mathsf{pk}_{sig})$
    Parse $\sigma^*$ as $(\sigma_{\text{FIX}},\sigma',\text{ADM},\mathsf{pk}',c,\tau)$.
    $m^*_{\text{FIX}} := (\text{FIX}_{\text{ADM}}(m),\text{ADM},\mathsf{pk}^*_{san})$
    output $(m^*_{\text{FIX}},\sigma^*_{\text{FIX}})$
```

```
$\mathsf{Sign}'(m,\mathsf{pk}_{san},\text{ADM})$ :
    $\rho \leftarrow \chi$
    $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk},\rho)$
    $\mathsf{sk}' \leftarrow \mathsf{RandPK}(\mathsf{sk},\rho)$
    $c \leftarrow \mathsf{Enc}(\mathsf{ek},\mathsf{pk};\omega)$
    $x := (c,\mathsf{ek},\mathsf{pk},\mathsf{pk}_{san},\mathsf{pk}')$
    $\tau \leftarrow \mathsf{P}_{\mathsf{PoK}}(\mathsf{crs}_{\mathsf{PoK}},x,(\rho,\omega))$
    $m_{\text{FIX}} := (\text{FIX}_{\text{ADM}}(m),\text{ADM},\mathsf{pk}_{san})$
    $\sigma_{\text{FIX}} \leftarrow \mathcal{O}(m_{\text{FIX}})$
    $\sigma' \leftarrow \mathsf{SSign}(\mathsf{sk}',m,\rho)$
    output $\sigma = (\sigma_{\text{FIX}},\sigma',\text{ADM},\mathsf{pk}',c,\tau)$
```

Fig. 3. Description of reduction $\mathcal{B}_3$, reducing the immutability of SanS against the s-EUF security of Σ_{FIX}.

Observe that this reduction is clearly efficient and perfectly simulates the view of $\mathcal{A}$ in the game $\mathsf{Immut}^{\text{SanS}}_{\mathcal{A}}(\kappa)$. Furthermore, because of Equation 22, m^*_{FIX} is a message never queried to the signing oracle. It therefore holds that

$$\Pr\left[\mathsf{s\text{-}EUF}^{\Sigma_{\text{FIX}}}_{\mathcal{B}_3}(\kappa)=1\right] \geq \Pr\left[\mathsf{Immut}^{\text{SanS}}_{\mathcal{A}}(\kappa)=1\right]$$

which must be negligible because the signature scheme is strongly existentially unforgeable.

Thus it must hold that $\Pr\left[\mathsf{Immut}^{\text{SanS}}_{\mathcal{A}}(\kappa)=1\right]$ is negligible.

Theorem 6 ((Proof-Restricted) Transparency). *If $\Pi_{PoK} = (\mathsf{Setup}_{\mathsf{PoK}},\mathsf{P}_{\mathsf{PoK}},\mathsf{V}_{\mathsf{PoK}})$ is a computationally zero-knowledge perfectly sound proof of knowledge system, $\Pi_{ZK} = (\mathsf{Setup}_{\mathsf{ZK}},\mathsf{P}_{\mathsf{ZK}},\mathsf{V}_{\mathsf{ZK}})$ is a computationally zero-knowledge proof system, $\mathcal{E} = (\mathsf{EGen},\mathsf{Enc},\mathsf{Dec})$ is a CCA-secure public key encryption scheme, and $\Sigma = (\mathsf{SSetup},\mathsf{SGen},\mathsf{SSign},\mathsf{SVerify},\mathsf{RandSK},\mathsf{RandPK})$ is a signature scheme with perfectly re-randomizable keys that is unforgeable under re-randomized keys, then* SanS *is (proof-restrictedly) transparent.*

Proof. We use a series of games to prove that the two cases of the $\mathsf{Trans}^{\mathcal{A}}_{\text{SanS}}(\kappa)$ are indistinguishable for any polynomial time adversary $\mathcal{A}$.

Game_0 is exactly the $\mathsf{Trans}^{\mathcal{A}}_{\text{SanS}}(\kappa)$ experiment with b fixed to 1.

Game_1 works exactly as Game_0, except that $\mathsf{crs}_{\mathsf{PoK}}$ is chosen as $(\mathsf{crs}_{\mathsf{PoK}},\mathrm{T}_{\mathrm{PoK}}) \leftarrow \mathsf{S}_{\mathsf{PoK},0}(1^\kappa)$ and the proofs τ in the answers to $\mathsf{Sanit}/\mathsf{Sign}$ queries are computed as $\tau \leftarrow \mathsf{S}_{\mathsf{PoK},1}(\mathsf{crs},\mathrm{T}_{\mathrm{PoK}},x)$, where $\mathsf{S}_{\mathsf{PoK}} = (\mathsf{S}_{\mathsf{PoK},0},\mathsf{S}_{\mathsf{PoK},1})$ is the simulator of Π_{PoK}.

Game_2 works exactly as Game_1, except that $\mathsf{crs}_{\mathsf{ZK}}$ is chosen as $(\mathsf{crs}_{\mathsf{ZK}},\mathrm{T}_{\mathrm{ZK}}) \leftarrow \mathsf{S}_{\mathsf{ZK},0}(1^\kappa)$ and the proofs of decryption ϕ in the answers to Proof queries are computed as $\phi \leftarrow \mathsf{S}_{\mathsf{ZK},1}(\mathsf{crs},\mathrm{T}_{\mathrm{ZK}},x)$, where $\mathsf{S}_{\mathsf{ZK}} = (\mathsf{S}_{\mathsf{ZK},0},\mathsf{S}_{\mathsf{ZK},1})$ is the simulator of Π_{ZK}.

Game_3 works exactly as Game_2, except for the following changes. The ciphertexts c in the answers to $\underline{\mathsf{Sanit}/\mathsf{Sign}}$ queries are computed as $c \leftarrow \mathsf{Enc}(\mathsf{ek},\overline{\mathsf{pk}})$ for an independently chosen but fixed public key $\overline{\mathsf{pk}}$. Let $C_{\mathsf{Sanit}/\mathsf{Sign}}$ be the set of ciphertexts computed this way. For ciphertexts $c \notin C_{\mathsf{Sanit}/\mathsf{Sign}}$, the Proof oracle proceeds exactly as in the previous game. For ciphertexts $c \in C_{\mathsf{Sanit}/\mathsf{Sign}}$ however, the Proof oracle sets $\widehat{\mathsf{pk}} := \mathsf{pk}$ instead of decrypting c, before proceeding as before.

Game_4 works exactly as Game_3, except that the bit b is fixed to 0 and the Proof oracle sets $\widehat{\mathsf{pk}} := \mathsf{pk}_{san}$ for ciphertexts $c \in C_{\mathsf{Sanit}/\mathsf{Sign}}$.

Game_5 works exactly as Game_4, except that the ciphertext c in the answers to queries to the $\mathsf{Sanit}/\mathsf{Sign}$ oracle is computed as $c \leftarrow \mathsf{Enc}(\mathsf{ek},\mathsf{pk}_{san})$ and the Proof oracle again always uses decryption to determine $\widehat{\mathsf{pk}}$.

Game_6 works exactly as Game_5, except that $\mathsf{crs}_{\mathsf{ZK}}$ is once again chosen honestly as $\mathsf{crs}_{\mathsf{ZK}} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^\kappa)$ and the proofs of decryption ϕ in the answers to Proof queries are computed honestly again as $\phi \leftarrow \mathsf{P}_{\mathsf{ZK}}(\mathsf{crs}_{\mathsf{ZK}},x,(\psi,\mathsf{dk}))$.

Game_7 works exactly as Game_6, except that $\mathsf{crs}_{\mathsf{PoK}}$ is once again chosen honestly as $\mathsf{crs}_{\mathsf{PoK}} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^\kappa)$ and the proofs τ in the answers to $\mathsf{Sanit/Sign}$ queries are computed honestly as $\tau \leftarrow \mathsf{P}_{\mathsf{PoK}}(\mathsf{crs}_{\mathsf{PoK}}, x, (\rho, \omega))$. This is exactly the $\mathsf{Trans}^{\mathcal{A}}_{\mathrm{SanS}}(\kappa)$ experiment with b fixed to 0.

We argue that each pair of neighboring games cannot be distinguished, except with negligible probability, by a probabilistic polynomial time adversary.

$\underline{\mathsf{Game}_0 \approx \mathsf{Game}_1}$ Let $\mathcal{A}$ be a probabilistic polynomial time adversary distinguishing Game_0 and Game_1 with probability $1/2+\epsilon(\kappa)$. Now, consider reduction $\mathcal{B}_4$, depicted in Figure 4 against the zero-knowledge property of the underlying proof of knowledge system.

```
$\underline{\mathcal{B}_4^{\mathcal{O}(\cdot,\cdot)}(\mathsf{crs}_{\mathsf{PoK}})}$ :
  $\mathsf{crs}_{\mathsf{ZK}} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^\kappa)$
  $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\kappa)$
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$
  $(\mathsf{sk}_{\mathrm{FIX}}, \mathsf{pk}_{\mathrm{FIX}}) \leftarrow \mathsf{SGen}_{\mathrm{FIX}}(1^\kappa)$
  $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{EGen}(1^\kappa; \psi)$
  $\mathsf{sk}_{sig} := (\mathsf{sk}_{\mathrm{FIX}}, \mathsf{sk}, \mathsf{dk}, \mathsf{pk}_{\mathrm{FIX}}, \mathsf{pk}, \mathsf{ek}, \psi)$
  $\mathsf{pk}_{sig} := (\mathsf{pk}_{\mathrm{FIX}}, \mathsf{pk}, \mathsf{ek})$
  $(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{SGen}(1^\kappa)$
  $a \leftarrow \mathcal{A}^{\substack{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_{san}),\\ \mathsf{Proof}(\mathsf{sk}_{sig},\cdot,\cdot,\cdot),\mathsf{Sanit/Sign}'(\cdot,\cdot,\cdot)}}(\mathsf{pk}_{sig}, \mathsf{pk}_{san})$
  output $a$
```

```
$\underline{\mathsf{Sanit/Sign}'(m, \mathrm{MOD}, \mathrm{ADM})}$ :
  If $\mathrm{ADM}(\mathrm{MOD}) = 0$
    output $\perp$
  $\rho \leftarrow \chi$
  $\mathsf{sk}' \leftarrow \mathsf{RandSK}(\mathsf{sk}_{sig}, \rho)$
  $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}_{sig}, \rho)$
  $m_{\mathrm{FIX}} := (\mathrm{FIX}_{\mathrm{ADM}}(m), \mathrm{ADM}, \mathsf{pk}_{san})$
  $\sigma_{\mathrm{FIX}} := \mathsf{SSign}_{\mathrm{FIX}}(\mathsf{sk}_{\mathrm{FIX}}, m_{\mathrm{FIX}})$
  $m' := \mathrm{MOD}(m)$
  $\sigma' := \mathsf{SSign}(\mathsf{sk}', m')$
  $c \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{pk})$
  $x := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}')$
  $\tau \leftarrow \mathcal{O}(x, (\rho, \omega))$
  $\sigma := (\sigma_{\mathrm{FIX}}, \sigma', \mathrm{ADM}, \mathsf{pk}', c, \tau)$
  return $(m', \sigma)$
```

Fig. 4. Reduction of the indistinguishability of Game_0 and Game_1 in the transparency proof to the zero-knowledge property of the underlying proof system Π_{PoK}.

Observe that this reduction is clearly efficient and perfectly simulates the view of $\mathcal{A}$ in the Game_0 if the oracle of $\mathcal{B}_4$ is the honest prover and in Game_1 if the oracle of $\mathcal{B}_4$ is the simulator. It thus follows immediately

$$\left| \begin{array}{l} \Pr\left[\mathsf{crs}_{\mathsf{PoK}} \leftarrow \mathsf{Setup}_{\mathsf{PoK}}(1^\kappa) : \mathcal{B}_4^{\mathsf{P}_{\mathsf{PoK}}(\mathsf{crs}_{\mathsf{PoK}},\cdot,\cdot)}(\mathsf{crs}_{\mathsf{PoK}}) = 1\right] \\ - \Pr\left[(\mathsf{crs}_{\mathsf{PoK}}, \mathrm{T}_{\mathrm{PoK}}) \leftarrow \mathsf{S}_{\mathsf{PoK},0}(1^\kappa) : \mathcal{B}_4^{\mathsf{S}'(\mathsf{crs}_{\mathsf{PoK}},\mathrm{T}_{\mathrm{PoK}},\cdot,\cdot)}(\mathsf{crs}_{\mathsf{PoK}}) = 1\right] \end{array} \right| = \epsilon(\kappa).$$

Therefore $\epsilon(\kappa)$ must be negligible, because Π_{PoK} is zero knowledge.

$\underline{\mathsf{Game}_1 \approx \mathsf{Game}_2}$ Let $\mathcal{A}$ be a probabilistic polynomial time adversary distinguishing Game_1 and Game_2 with probability $1/2+\epsilon(\kappa)$. Now, consider reduction $\mathcal{B}_5$, depicted in Figure 5 against the zero-knowledge property of the underlying non-interactive zero knowledge proof system.

Observe that this reduction is clearly efficient and perfectly simulates the view of $\mathcal{A}$ in the Game_1 if the oracle of $\mathcal{B}_5$ is the honest prover and in Game_2 if the oracle of $\mathcal{B}_5$ is the simulator. It thus follows immediately

$$\left| \begin{array}{l} \Pr\left[\mathsf{crs}_{\mathsf{ZK}} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^\kappa) : \mathcal{B}_5^{\mathsf{P}_{\mathsf{ZK}}(\mathsf{crs}_{\mathsf{ZK}},\cdot,\cdot)}(\mathsf{crs}_{\mathsf{ZK}}) = 1\right] \\ - \Pr\left[(\mathsf{crs}_{\mathsf{ZK}}, \mathrm{T}_{\mathrm{ZK}}) \leftarrow \mathsf{S}_{\mathsf{ZK},0}(1^\kappa) : \mathcal{B}_5^{\mathsf{S}'(\mathsf{crs}_{\mathsf{ZK}},\mathrm{T}_{\mathrm{ZK}},\cdot,\cdot)}(\mathsf{crs}_{\mathsf{ZK}}) = 1\right] \end{array} \right| = \epsilon(\kappa).$$

Therefore $\epsilon(\kappa)$ must be negligible, because Π_{PoK} is zero knowledge.

$\underline{\mathsf{Game}_2 \approx \mathsf{Game}_3}$ Let $\mathcal{A}$ be a probabilistic polynomial time adversary distinguishing Game_2 and Game_3 with probability $1/2 + \epsilon(\kappa)$. Now, consider reduction $\mathcal{B}_6$, depicted in Figure 6 against the CCA security of the underlying encryption scheme.

Note, that we reduce to a variant of CCA security, where the adversary can send multiple challenges to an oracle $\mathcal{O}$. The decryption oracle will not answer any queries made up of ciphertexts output by $\mathcal{O}$. This variant of CCA security follows from standard CCA security by a standard hybrid argument. Observe that this reduction is clearly efficient Further, if the bit chosen by the IND-CCA experiment is 0, then $\mathcal{B}_6$ perfectly simulates Game_2. The only place where the reduction deviates from the exact behavior of Game_2 is in answering Proof queries for ciphertexts in $C_{\mathsf{Sanit/Sign}}$. However, even in those cases, the "decryption" is in fact correct, and since the proof of decryption is simulated, the fact that the witness is not known does not change the distribution of the answer.

If the bit chosen by the CCA experiment is 1, then $\mathcal{B}_6$ perfectly simulates Game_3. It thus follows that

$$\Pr\left[\mathsf{IND\text{-}CCA}^{\mathcal{E}}_{\mathcal{B}_6^{\mathcal{A}}}(\kappa) = 1\right] \leq \frac{1}{2} + \epsilon(\kappa)$$

Therefore $\epsilon(\kappa)$ must be negligible, because $\mathcal{E}$ is CCA secure.

$\underline{\mathsf{Game}_3 \approx \mathsf{Game}_4}$ The only differences between the two games are the way in which queries to $\mathsf{Sanit/Sign}$ and Proof oracles are answered. In the case of the $\mathsf{Sanit/Sign}$ oracle, the only difference is, that in Game_3 the signer's key is re-randomized and in Game_4, the sanitizer's key is re-randomized. However, by virtue of the perfect re-randomizability property of the signature scheme, the re-randomized keys are in fact distributed identically in both cases. Further, the remainder of the signature is computed independently from the re-randomization factor ρ due to the simulation of the proof τ. Therefore, the outputs of $\mathsf{Sanit/Sign}$ are distributed identically in both cases.

We denote by $S_{\mathsf{Sanit/Sign}}$ the sets of signatures output as answers by the $\mathsf{Sanit/Sign}$ oracle. In the case of the Proof oracle, there is only a difference, if the attacker makes a valid query $(m, \sigma = (\sigma_{\text{FIX}}, \sigma', \text{ADM}, \mathsf{pk}', c, \tau), \mathsf{pk}'_{san})$ such that the following conditions hold.

$$\mathsf{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}'_{san}) = 1 \tag{23}$$

$$\exists(\sigma_{\text{FIX},i}, \sigma'_i, \text{ADM}_i, \mathsf{pk}'_i, c_i, \tau_i) \in S_{\mathsf{Sanit/Sign}} : c = c_i, \tag{24}$$

Let query denote the event that such a query happens. We can split the probability of query occurs as follows:

$$\Pr[\mathsf{query}] = \Pr\left[\mathsf{query} \wedge \mathsf{pk}'_{san} \neq \mathsf{pk}_{san}\right] + \Pr\left[\mathsf{query} \wedge \mathsf{pk}'_{san} = \mathsf{pk}_{san}\right].$$

```
$\underline{\mathcal{B}_5^{\mathcal{O}(\cdot,\cdot)}(\mathsf{crs}_{\mathsf{ZK}})}$ :
  $(\mathsf{crs}_{\mathsf{PoK}}, \mathrm{T}_{\mathrm{PoK}}) \leftarrow \mathsf{S}_{\mathsf{PoK},0}(1^\kappa)$
  $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\kappa)$
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$
  $(\mathsf{sk}_{\text{FIX}}, \mathsf{pk}_{\text{FIX}}) \leftarrow \mathsf{SGen}_{\text{FIX}}(1^\kappa)$
  $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{EGen}(1^\kappa; \psi)$
  $\mathsf{sk}_{sig} := (\mathsf{sk}_{\text{FIX}}, \mathsf{sk}, \mathsf{dk}, \mathsf{pk}_{\text{FIX}}, \mathsf{pk}, \mathsf{ek}, \psi)$
  $\mathsf{pk}_{sig} := (\mathsf{pk}_{\text{FIX}}, \mathsf{pk}, \mathsf{ek})$
  $(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{SGen}(1^\kappa)$
  $a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_{san}),\mathsf{Proof}'(\cdot,\cdot,\cdot),\mathsf{Sanit/Sign}(\cdot,\cdot,\cdot)}(\mathsf{pk}_{sig}, \mathsf{pk}_{san})$
  output $a$
```

```
$\underline{\mathsf{Proof}'(m, \sigma, \mathsf{pk}_{san})}$ :
  If $\mathsf{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 0$
    return $\perp$
  Extract $c$ from $\sigma$.
  $\widehat{\mathsf{pk}} \leftarrow \mathsf{Dec}(\mathsf{dk}, c)$
  $x := (\mathsf{ek}, c, \widehat{\mathsf{pk}})$
  $\phi \leftarrow \mathcal{O}(x, (\psi, \mathsf{dk}))$
  output $(\widehat{\mathsf{pk}}, \phi)$
```

Fig. 5. Reduction of the indistinguishability of Game_1 and Game_2 in the transparency proof to the zero-knowledge property of the underlying proof system Π_{ZK}.

```
$\underline{\mathcal{B}_6^{\mathsf{Dec}(\mathsf{dk},\cdot),\mathcal{O}(\cdot,\cdot)}(\mathsf{ek})}$ :
  $(\mathsf{crs}_{\mathsf{PoK}}, \mathrm{T}_{\mathrm{PoK}}) \leftarrow \mathsf{S}_{\mathsf{PoK},0}(1^\kappa)$
  $(\mathsf{crs}_{\mathsf{ZK}}, \mathrm{T}_{\mathrm{ZK}}) \leftarrow \mathsf{S}_{\mathsf{ZK},0}(1^\kappa)$
  $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\kappa)$
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$
  $(\mathsf{sk}_{\mathrm{FIX}}, \mathsf{pk}_{\mathrm{FIX}}) \leftarrow \mathsf{SGen}_{\mathrm{FIX}}(1^\kappa)$
  $\mathsf{sk}_{sig} := (\mathsf{sk}_{\mathrm{FIX}}, \mathsf{sk}, ?, \mathsf{pk}_{\mathrm{FIX}}, \mathsf{pk}, \mathsf{ek}, ?)$
  $\mathsf{pk}_{sig} := (\mathsf{pk}_{\mathrm{FIX}}, \mathsf{pk}, \mathsf{ek})$
  $(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{SGen}(1^\kappa)$
  $(\overline{\mathsf{sk}}, \overline{\mathsf{pk}}) \leftarrow \mathsf{SGen}(1^\kappa)$
  $C_{\mathsf{Sanit/Sign}} := \emptyset$
  $a \leftarrow \mathcal{A}^{\substack{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_{san}),\\ \mathsf{Proof}'(\cdot,\cdot,\cdot),\mathsf{Sanit/Sign}'(\cdot,\cdot,\cdot)}}(\mathsf{pk}_{sig}, \mathsf{pk}_{san})$
  Output $a$
```

```
$\underline{\mathsf{Sanit/Sign}'(m, \textsc{Mod}, \textsc{Adm})}$ :
  $\rho \leftarrow \chi$
  $\mathsf{sk}' \leftarrow \mathsf{RandSK}(\mathsf{sk}, \rho)$
  $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}, \rho)$
  $\sigma_{\mathrm{FIX}} := \mathsf{SSign}_{\mathrm{FIX}}(\mathsf{sk}_{\mathrm{FIX}}, m_{\mathrm{FIX}})$
  $m' := \textsc{Mod}(m)$
  $\sigma' := \mathsf{SSign}(\mathsf{sk}', m')$
  $c \leftarrow \mathcal{O}(\mathsf{pk}, \overline{\mathsf{pk}})$
  $C_{\mathsf{Sanit/Sign}} := C_{\mathsf{Sanit/Sign}} \cup \{c\}$
  $x := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}')$
  $\tau \leftarrow \mathsf{S}_{\mathsf{PoK},1}(\mathsf{crs}_{\mathsf{PoK}}, \mathrm{T}_{\mathrm{PoK}}, x)$
  $\sigma := (\sigma_{\mathrm{FIX}}, \sigma', \textsc{Adm}, \mathsf{pk}', c, \tau)$
  return $(m', \sigma)$
```

```
$\underline{\mathsf{Proof}'(m, \sigma, \mathsf{pk}_{san})}$ :
  If $\mathsf{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 0$
    return $\perp$
  Extract $c$ from $\sigma$.
  If $c \in C_{\mathsf{Sanit/Sign}}$
    $\widehat{\mathsf{pk}} := \mathsf{pk}$
  else
    $\widehat{\mathsf{pk}} \leftarrow \mathsf{Dec}(c)$
  $x := (\mathsf{ek}, c, \widehat{\mathsf{pk}})$
  $\phi \leftarrow \mathsf{S}_{\mathsf{ZK},(}\mathsf{crs}_{\mathsf{ZK}}, x)$
  output $(\widehat{\mathsf{pk}}, \phi)$
```

Fig. 6. Reduction of the indistinguishability of Game_2 and Game_3 to the CCA security of the underlying encryption scheme.

Note that in the first case $\mathcal{A}$ must compute a new proof τ, such that

$$\mathsf{V}_{\mathsf{PoK}}(\mathsf{crs}_{\mathsf{PoK}}, (\mathsf{ek}, c, \mathsf{pk}, \mathsf{pk}'_{san}, \mathsf{pk}'), \tau) = 1.$$

Since c is an encryption of $\overline{\mathsf{pk}}$, and $\mathsf{pk} \neq \overline{\mathsf{pk}}$ except with negligible probability, the perfect soundness of $\mathit{\Pi}_{PoK}$ implies that $\mathsf{pk}'_{san} = \overline{\mathsf{pk}}$. This leads to a trivial reduction to the CCA security (even one-wayness) of the encryption scheme $\mathcal{E}$.

In the second case, we can reduce to the UFRK security of the signature scheme $\mathit{\Sigma}$ as depicted in Figure 7.

```
$\underline{\mathcal{B}_8^{\mathcal{O}_1(\mathsf{sk},\cdot),\mathcal{O}_2(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk}_{\mathsf{UFRK}})}$ :
  $(\mathsf{crs}_{\mathsf{PoK}}, \mathrm{T}_{\mathrm{PoK}}) \leftarrow \mathsf{S}_{\mathsf{PoK},0}(1^\kappa)$
  $(\mathsf{crs}_{\mathsf{ZK}}, \mathrm{T}_{\mathrm{ZK}}) \leftarrow \mathsf{S}_{\mathsf{ZK},0}(1^\kappa)$
  $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\kappa)$
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$
  $(\mathsf{sk}_{\mathrm{FIX}}, \mathsf{pk}_{\mathrm{FIX}}) \leftarrow \mathsf{SGen}_{\mathrm{FIX}}(1^\kappa)$
  $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{EGen}(1^\kappa; \psi)$
  $\mathsf{sk}_{sig} := (\mathsf{sk}_{\mathrm{FIX}}, \mathsf{sk}, \mathsf{dk}, \mathsf{pk}_{\mathrm{FIX}}, \mathsf{pk}, \mathsf{ek}, \psi)$
  $\mathsf{pk}_{sig} := (\mathsf{pk}_{\mathrm{FIX}}, \mathsf{pk}, \mathsf{ek})$
  $(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{SGen}(1^\kappa)$
  $(\overline{\mathsf{sk}}, \overline{\mathsf{pk}}) \leftarrow \mathsf{SGen}(1^\kappa)$
  $C_{\mathsf{Sanit/Sign}} := \emptyset$
  $\mathcal{A}^{\substack{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_{san}),\\ \mathsf{Proof}'(\cdot,\cdot,\cdot),\mathsf{Sanit/Sign}'(\cdot,\cdot,\cdot)}}(\mathsf{pk}_{sig}, \mathsf{pk}_{san})$
```

```
$\underline{\mathsf{Sanit/Sign}'(m, \textsc{Mod}, \textsc{Adm})}$ :
  $\rho \leftarrow \chi$
  $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}_{\mathsf{UFRK}}, \rho)$
  $\sigma_{\mathrm{FIX}} := \mathsf{SSign}_{\mathrm{FIX}}(\mathsf{sk}_{\mathrm{FIX}}, m_{\mathrm{FIX}})$
  $m' := \textsc{Mod}(m)$
  $\sigma' := \mathcal{O}_2(m', \rho)$
  $c \leftarrow \mathsf{Enc}(\mathsf{ek}, \overline{\mathsf{pk}})$
  $C_{\mathsf{Sanit/Sign}} := C_{\mathsf{Sanit/Sign}} \cup \{(c, \rho)\}$
  $x := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}')$
  $\tau \leftarrow \mathsf{S}_{\mathsf{PoK},1}(\mathsf{crs}_{\mathsf{PoK}}, \mathrm{T}_{\mathrm{PoK}}, x)$
  $\sigma := (\sigma_{\mathrm{FIX}}, \sigma', \textsc{Adm}, \mathsf{pk}', c, \tau)$
  return $(m', \sigma)$
```

```
$\underline{\mathsf{Proof}'(m, \sigma, \mathsf{pk}'_{san})}$ :
  If $\mathsf{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}'_{san}) = 0$
    return $\perp$
  Parse $\sigma$ as $(\sigma_{\mathrm{FIX}}, \sigma', \textsc{Adm}, \mathsf{pk}', c, \tau)$.
  If $(c_i, \rho_i) \in C_{\mathsf{Sanit/Sign}}$ with $c_i = c$
    abort reduction, and output
      $(m, \sigma', \rho_i)$ as forgery
  $\widehat{\mathsf{pk}} \leftarrow \mathsf{Dec}(c)$
  $x := (\mathsf{ek}, c, \widehat{\mathsf{pk}})$
  $\phi \leftarrow \mathsf{S}_{\mathsf{ZK},(}\mathsf{crs}_{\mathsf{ZK}}, x)$
  output $(\widehat{\mathsf{pk}}, \phi)$
```

Fig. 7. Reduction of the indistinguishability of Game_3 and Game_4 in the case where $\mathsf{pk}'_{san} = \mathsf{pk}_{san}$ to the UFRK security of the underlying signature scheme.

Since the reduction only runs a constant number of polynomial time bounded algorithms, the reduction $\mathcal{B}_8$ is clearly efficient.

Further, it perfectly simulates both games up until a Proof query is made satisfying Equation 23 and Equation 24. Once such a query is made, the reduction outputs (m, σ', ρ_i) as a forgery. The definition of transparency guarantees that m is a new message that has not been queried to the UFRK signing oracle before. The fact that $\mathsf{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}'_{san}) = 1$ guarantees that $\mathsf{SVerify}(\mathsf{pk}', m, \sigma') = 1$ and that $\mathsf{V}_{\mathsf{PoK}}(\mathsf{crs}_{\mathsf{PoK}}, (\mathsf{ek}, c, \mathsf{pk}, \mathsf{pk}'_{san}, \mathsf{pk}'), \tau) = 1)$. In the case where $\mathsf{pk}'_{san} = \mathsf{pk}_{san}$, it holds that $(\mathsf{ek}, c, \mathsf{pk}, \mathsf{pk}'_{san}, \mathsf{pk}') \notin \mathcal{L}_1$. Therefore, due to the perfect soundness, $\mathcal{A}$ cannot compute τ for a new statement of this form. This implies that

$$\mathsf{pk}' = \mathsf{RandPK}(\mathsf{pk}_{\mathsf{UFRK}}, \rho_i),$$

and therefore (m, σ', ρ_i) is a valid forgery.

It thus follows that

$$\begin{aligned}\Pr[\mathsf{query}] &= \Pr[\mathsf{query} \wedge \tau \neq \tau_i] + \Pr[\mathsf{query} \wedge \tau = \tau_i] \\ &\leq \left(\Pr\left[\mathsf{IND\text{-}CCA}^{\mathcal{E}}_{\mathcal{B}_7^{\mathcal{A}}}(\kappa)\right] - \frac{1}{2}\right) + \Pr\left[\mathsf{UFRK}^{\mathcal{E}}_{\mathcal{B}_8^{\mathcal{A}}}(\kappa)\right]\end{aligned}$$

and therefore query happens only with negligible probability and Game_3 and Game_4 are thus indistinguishable.

$\underline{\mathsf{Game}_4 \approx \mathsf{Game}_5}$ This hop is completely symmetrical to the hop between Game_2 to Game_3. The reduction therefore also works almost identically. The only difference being that the sanitizer's key is randomized instead of the signer's when answering $\mathsf{Sanit}/\mathsf{Sign}$ queries and and the Proof' oracle sets $\widehat{\mathsf{pk}} := \mathsf{pk}_{san}$ for ciphertexts $c \in C_{\mathsf{Sanit}/\mathsf{Sign}}$.

$\underline{\mathsf{Game}_5 \approx \mathsf{Game}_6}$ This hop essentially reverts the changes made in the hop from Game_1 to Game_2. The reduction therefore also works almost identically. The only difference being that the sanitizer's key is randomized instead of the signer's when answering $\mathsf{Sanit}/\mathsf{Sign}$ queries.

$\underline{\mathsf{Game}_6 \approx \mathsf{Game}_7}$ Just as in the hop before, this hop essentially reverts the changes made in the hop from Game_0 to Game_1. The reduction is once again almost identically, the difference being that the sanitizer's key is randomized instead of the signer's when answering $\mathsf{Sanit}/\mathsf{Sign}$ queries.

Since the distinguishing advantage of an probabilistic polynomial time attacker is negligible for each step, it follows by a simple union bound that the two cases of $\mathsf{Trans}^{\mathcal{A}}_{\mathrm{SanS}}(\kappa)$ with $b = 1$ and $b = 0$ are also indistinguishable and thus SanS is proof-restrictedly transparent.

Theorem 7 (Unlinkability). *If the deterministic signature scheme* $\Sigma_{\mathrm{FIX}} = (\mathsf{SSetup}_{\mathrm{FIX}}, \mathsf{SGen}_{\mathrm{FIX}}, \mathsf{SSign}_{\mathrm{FIX}}, \mathsf{SVerify}_{\mathrm{FIX}})$ *is is strongly existentially unforgeable, then the construction given in Section 4 is unlinkable.*

Proof. Let $\mathcal{A}$ be a probabilistic polynomial time adversary against the signer unlinkability of SanS. Let

$$((m_i^0, \textsc{Mod}_i^0, \sigma_i^0), (m_i^1, \textsc{Mod}_i^1, \sigma_i^1))$$

be a query made by $\mathcal{A}$ to the $\mathsf{LoRSanit}$ oracle, where σ_i^b can be parsed as $(\sigma^b_{\mathrm{FIX},i}, {\sigma'}^b_i, \textsc{Adm}^b_i, {\mathsf{pk}'}^b_i, c^b_i, \tau^b_i)$. The only difference between the two cases of the unlinkability experiment is the distribution of the answers to these queries if it holds that

$$\mathsf{Verify}(m_i^0, \sigma_i^0, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 1 \tag{25}$$

$$\mathsf{Verify}(m_i^1, \sigma_i^1, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 1 \tag{26}$$

$$\textsc{Adm}_i^0(\textsc{Mod}_i^0) \neq 0 \tag{27}$$

$$\textsc{Adm}_i^1(\textsc{Mod}_i^1) \neq 0 \tag{28}$$

$$\textsc{Mod}_i^0(m_i^0) = \textsc{Mod}_i^1(m_i^1) \tag{29}$$

$$\textsc{Adm}_i^0 = \textsc{Adm}_i^1 \tag{30}$$

Let (m^*_b, σ^*_b) denote the answer to such a query depending on the choice of b in the experiment, where σ^*_b can be parsed as $(\sigma_{\text{FIX},b}, \sigma'_b, \text{ADM}_b, \mathsf{pk}'_b, c_b, \tau_b)$.

From Equation 30 it follows directly that

$$\text{ADM}_0 = \text{ADM}_1 \tag{31}$$

Further, it follows from this, the definition of Sanit, and the perfect re-randomizability of the signature scheme Σ that the the distributions

$$(\sigma'_0, \text{ADM}_0, \mathsf{pk}'_0, c_0, \tau_0) \sim (\sigma'_1, \text{ADM}_1, \mathsf{pk}'_1, c_1, \tau_1) \tag{32}$$

are identical.

From Equation 30 it follows by the uniqueness of FIX_{ADM}, that

$$\text{FIX}_{\text{ADM}^0_i}(m^0_i, \text{ADM}^0, \mathsf{pk}_{san}) = \text{FIX}_{\text{ADM}^1_i}(m^1_i, \text{ADM}^1, \mathsf{pk}_{san}). \tag{33}$$

It holds by Equation 32, that the view of $\mathcal{A}$ only differs in the two cases of the unlinkability experiment, if it makes a query to $\mathsf{LoRSanit}$ such that in addition to Equation 25 through Equation 30 it holds that

$$\sigma^0_{\text{FIX},i} \neq \sigma^1_{\text{FIX},i} \tag{34}$$

We denote by query the event that such a query happens and thus get

$$\Pr\left[\mathsf{Link}^{\text{SanS}}_{\mathcal{A}}(\kappa) = 1\right] = \frac{1}{2} + \Pr\left[\mathsf{query}\right]$$

Now, consider reduction $\mathcal{B}_9$, depicted in Figure 8 against the strong existential unforgeability of the underlying signature scheme.

Observe that this reduction is clearly efficient and perfectly simulates the view of $\mathcal{A}$ in the game $\mathsf{Link}^{\text{SanS}}_{\mathcal{A}}(\kappa)$ unless query occurs. Whenever query occurs, it holds because of Equation 33 that $m^0_{\text{FIX}} = m^1_{\text{FIX}}$. Further, since Σ is deterministic, it holds that

$$\{(m^0_{\text{FIX}}, \sigma^0_{\text{FIX}}), (m^1_{\text{FIX}}, \sigma^1_{\text{FIX}})\} \cap L_\sigma \neq \emptyset.$$

Together with Equation 25 and Equation 26 it thus follows that

$$\Pr\left[\mathsf{query}\right] = \Pr[\mathsf{s\text{-}EUF}^{\Sigma}_{\mathcal{B}_9}(\kappa) = 1$$

and therefore

$$\Pr[\mathsf{Link}^{\text{SanS}}_{\mathcal{A}}(\kappa) = 1] = \frac{1}{2} + \Pr[\mathsf{s\text{-}EUF}^{\Sigma}_{\mathcal{B}_9}(\kappa) = 1],$$

where the second part of the sum must be negligible because the signature scheme is strongly existentially unforgeable.

Thus it must hold that $\Pr\left[\mathsf{Link}^{\text{SanS}}_{\mathcal{A}}(\kappa) = 1\right]$ is only negligibly greater than $1/2$.

5 Instantiating the Construction

We instantiate our generic construction with compatible and efficient instantiations in the random oracle model. For the two signature schemes, we choose standard Schnorr signatures as defined in Definition 11 for Σ, as well as a derandomized[2] version of Schnorr signatures for Σ_{FIX}[3]. The encryption scheme and proof systems are instantiated with the Cramer Shoup encryption scheme [CS98], and Σ-protocols that we convert into a non-interactive zero-knowledge proof via the Fiat-Shamir transform [FS87]. The Cramer Shoup encryption scheme is defined as follows:

[2] The randomness is generated by a PRF

[3] Note, that while the original security proof [PS96, PS00] for Schnorr signatures only proves standard existential unforgeability, it can be easily adapted to prove strong existential unforgeability

```
$\underline{\mathcal{B}_9^{\mathcal{O}(\mathsf{sk}_{\text{Fix}},\cdot)}(\mathsf{pk}_{\text{Fix}}):}$
  $(\mathsf{dk}, \mathsf{ek}, \psi) \leftarrow \mathsf{EGen}(1^\kappa; \psi)$
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$
  $\mathsf{pk}_{sig} := (\mathsf{pk}_{\text{Fix}}, \mathsf{pk}, \mathsf{ek})$
  $(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{SGen}(1^\kappa)$
  $L_\sigma = \emptyset$
  $b \leftarrow \{0,1\}$
  $\mathcal{A}^{\substack{\mathsf{Sign}'(\cdot,\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_{san}),\\ \mathsf{Proof}(\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{LoRSanit}'(\cdot,\cdot)}}(\mathsf{pk}_{sig}, \mathsf{pk}_{san})$
```

```
$\underline{\mathsf{Sign}'(m, \mathsf{pk}_{san}, \text{Adm}):}$
  $m_{\text{Fix}} := (\text{Fix}_{\text{Adm}}(m), \text{Adm}, \mathsf{pk}_{san})$
  $\sigma_{\text{Fix}} \leftarrow \mathcal{O}(m_{\text{Fix}})$
  $L_\sigma := L_\sigma \cup \{(m_{\text{Fix}}, \sigma_{\text{Fix}})\}$
  $\rho \leftarrow \chi$
  $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}, \rho)$
  $\mathsf{sk}' \leftarrow \mathsf{RandSK}(\mathsf{sk}, \rho)$
  $c \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{pk}; \omega)$
  $x := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}')$
  $\tau \leftarrow \mathsf{P}_{\mathsf{PoK}}(\mathsf{crs}_{\mathsf{PoK}}, x, (\rho, \omega))$
  output $(\sigma_{\text{Fix}}, \sigma', \text{Adm}, \mathsf{pk}', c, \tau)$
```

```
$\underline{\mathsf{LoRSanit}'((m^0, \text{Mod}^0, \sigma^0), (m^1, \text{Mod}^1, \sigma^1)):}$
  Parse $\sigma^0$ as $(\sigma^0_{\text{Fix}}, \sigma'^0, \text{Adm}^0, \mathsf{pk}'^0, c^0, \tau^0)$.
  Parse $\sigma^1$ as $(\sigma^1_{\text{Fix}}, \sigma'^1, \text{Adm}^1, \mathsf{pk}'^1, c^1, \tau^1)$.
  $m^0_{\text{Fix}} := (\text{Fix}^0_{\text{Adm}}(m), \text{Adm}^0, \mathsf{pk}_{san})$
  $m^1_{\text{Fix}} := (\text{Fix}^1_{\text{Adm}}(m), \text{Adm}^1, \mathsf{pk}_{san})$
  if $\mathsf{Verify}(m^0, \sigma^0, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 0$ or $\mathsf{Verify}(m^1, \sigma^1, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 0$
    or $\text{Adm}^0(\text{Mod}^0) = 0$ or $\text{Adm}^1(\text{Mod}^1) = 0$
    or $\text{Mod}^0(m^0) \neq \text{Mod}^1(m^1)$
    output $\perp$
  if $\sigma^0_{\text{Fix}} \neq \sigma^1_{\text{Fix}}$
    if $(m^0_{\text{Fix}}, \sigma^0_{\text{Fix}}) \notin L_\sigma$
      abort and output $(m^0_{\text{Fix}}, \sigma^0_{\text{Fix}})$
    else
      abort and output $(m^1_{\text{Fix}}, \sigma^1_{\text{Fix}})$
  $(m', \sigma') \leftarrow \mathsf{Sanit}(m^b, \text{Mod}^b, \sigma^b, \mathsf{pk}_{sig}, \mathsf{sk}_{san})$
  output $(m', \sigma')$
```

Fig. 8. Description of reduction $\mathcal{B}_9$, reducing the occurence of event query in the unlinkability experiment of SanS against the $\mathsf{s\text{-}EUF}$ security of Σ.

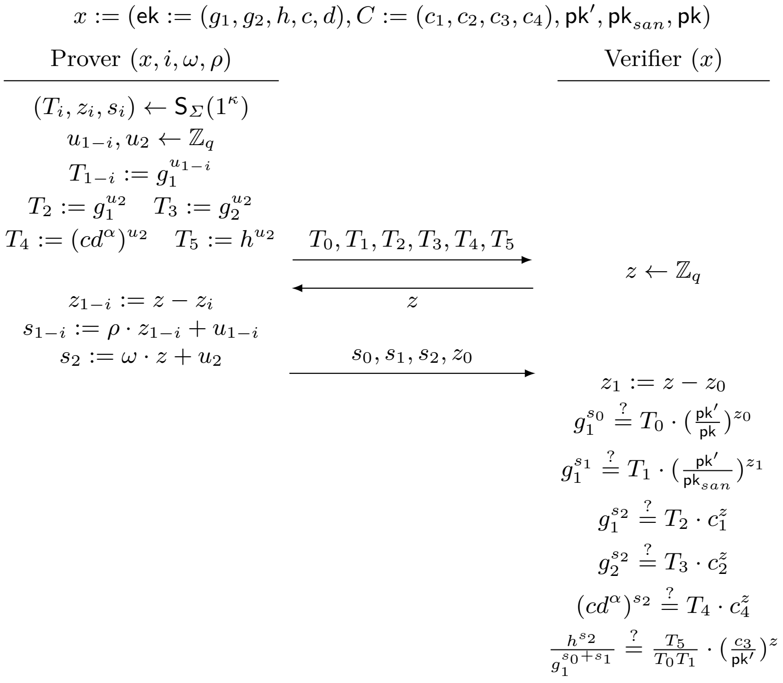
$x := (\mathsf{ek} := (g_1, g_2, h, c, d), C := (c_1, c_2, c_3, c_4), \mathsf{pk}', \mathsf{pk}_{san}, \mathsf{pk})$
Prover (x, i, ω, ρ)
Verifier (x)
$(T_i, z_i, s_i) \leftarrow \mathsf{S}_{\Sigma}(1^\kappa)$
$u_{1-i}, u_2 \leftarrow \mathbb{Z}_q$
$T_{1-i} := g_1^{u_{1-i}}$
$T_2 := g_1^{u_2}$ $T_3 := g_2^{u_2}$
$T_4 := (cd^\alpha)^{u_2}$ $T_5 := h^{u_2}$
$T_0, T_1, T_2, T_3, T_4, T_5$
$z \leftarrow \mathbb{Z}_q$
z
$z_{1-i} := z - z_i$
$s_{1-i} := \rho \cdot z_{1-i} + u_{1-i}$
$s_2 := \omega \cdot z + u_2$
s_0, s_1, s_2, z_0
$z_1 := z - z_0$
$g_1^{s_0} \overset{?}{=} T_0 \cdot (\frac{\mathsf{pk}'}{\mathsf{pk}})^{z_0}$
$g_1^{s_1} \overset{?}{=} T_1 \cdot (\frac{\mathsf{pk}'}{\mathsf{pk}_{san}})^{z_1}$
$g_1^{s_2} \overset{?}{=} T_2 \cdot c_1^z$
$g_2^{s_2} \overset{?}{=} T_3 \cdot c_2^z$
$(cd^\alpha)^{s_2} \overset{?}{=} T_4 \cdot c_4^z$
$\frac{h^{s_2}}{g_1^{s_0+s_1}} \overset{?}{=} \frac{T_5}{T_0 T_1} \cdot (\frac{c_3}{\mathsf{pk}'})^z$

Fig. 9. Σ-Protocol for Encryption of Public Key

Definition 21 (Cramer Shoup Encryption Scheme). *Let $\mathbb{G}$ be a cyclic group of prime order q with two random generators g_1, g_2 and let $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function. The Cramer Shoup encryption scheme, working over $\mathbb{G}$, is defined as follows:*

$\mathsf{EGen}(1^\kappa)$*: The key generation algorithm proceeds as follows: Pick $x, y, a, b, a', b' \leftarrow \mathbb{Z}_q$ uniformly at random, compute $h := g_1^x g_2^y$, $h := g_1^a g_2^b$, $h := g_1^{a'} g_2^{b'}$, set $\mathsf{dk} := (x, y, a, b, a', b')$ and $\mathsf{ek} := (h, c, d)$ and output $(\mathsf{dk}, \mathsf{ek})$.*

$\mathsf{Enc}(\mathsf{ek}, m)$*: The encryption algorithm proceeds as follows: Parse ek as (h, c, d) and choose $r \leftarrow \mathbb{Z}_q$ uniformly at random. Compute $\alpha := \mathcal{H}(g_1^r, g_2^r, h^r \cdot m)$ and $C := (g_1^r, g_2^r, h^r \cdot m, (cd^\alpha)^r)$. Output C.*

$\mathsf{Dec}(\mathsf{dk}, C)$*: The decryption algorithm proceeds as follows: Parse dk as (x, y, a, b, a', b') and C as (u, v, w, e). Compute $\alpha := \mathcal{H}(u, v, w)$ and check if $u^{a+\alpha a'} \cdot v^{b+ab'} = e$ holds. If it holds output $w/(u^x \cdot v^y)$. Otherwise output $\perp$.*

The remaining building blocks for our construction are two non-interactive zero-knowledge proof systems that we instantiate with specific Fiat-Shamir transformed [FS87] Σ-protocols. The first proof system is for the language $\mathcal{L}_1$ and the statement that we want to prove in our concrete instantiation looks as follows:

$$x := (\mathsf{ek} := (g_1, g_2, h, c, d), C := (c_1, c_2, c_3, c_4), \mathsf{pk}', \mathsf{pk}_{san}, \mathsf{pk})$$

$$PoK \left\{ (\omega, \rho) : \begin{array}{c} g_1^\omega = c_1 \ \wedge \ g_2^\omega = c_2 \ \wedge \ (cd^\alpha)^\omega = c_4 \\ \wedge \ \frac{h^\omega}{g^\rho} = \frac{c_3}{\mathsf{pk}'} \ \wedge \ \left(g_1^\rho = \frac{\mathsf{pk}'}{\mathsf{pk}} \ \vee \ g_1^\rho = \frac{\mathsf{pk}'}{\mathsf{pk}_{san}} \right) \end{array} \right\}.$$

Note that the statement that we are proving can be expressed as a logical combination of discrete logarithm proofs of knowledge. For the design of each single discrete logarithm proofs we deploy Schnorr's Σ-protocols from [Sch90]. We then formulate the complete proof using standard parallel composition techniques, first introduced in [CP93,CDS94]. The complete protocol is depicted in Figure 9. It is worth mentioning that, in order to express the logical disjunction of our statement, the prover must run the simulator S provided by the zero-knowledge property (Definition 19). For the specific case of Σ-protocols S_Σ works by randomly sampling z_i, s_i from $\mathbb{Z}_q$ and computing T_i as $g_1^{s_i}/(\frac{\mathsf{pk}'}{\mathsf{pk}})^{z_i}$ (or $g_1^{s_i}/(\frac{\mathsf{pk}'}{\mathsf{pk}_{san}})^{z_i}$, respectively). Finally, as mentioned above, the protocol can be made non-interactive by using the Fiat-Shamir transformation. Note that this allow us to drop the first tuple of elements $(T_0, \dots, T_5)$ since they can be simply recomputed from the public parameters and the further messages of the protocol and their integrity can be checked by recomputing the hash function.

In the following, we show how to instantiate the proof of knowledge for the language $\mathcal{L}_2$. We prove the following statement:

$$x := (\mathsf{ek} := (g_1, g_2, h, c, d), C := (c_1, c_2, c_3, c_4), \hat{\mathsf{pk}})$$

$$ZK\left\{(\chi, \psi) : g_1^{\chi} g_2^{\psi} = h \ \wedge \ c_1^{\chi} c_2^{\psi} = \frac{c_3}{\hat{\mathsf{pk}}}\right\}.$$

Again, for the concrete instantiation in Figure 10 we deploy parallel composition of Σ-protocols made non-interactive via the Fiat-Shamir transformation. Combining these building blocks yields a highly efficient sanitizable signature scheme.

$$x := (\mathsf{ek} := (g_1, g_2, h, c, d), C := (c_1, c_2, c_3, c_4), \hat{\mathsf{pk}})$$

$$\begin{array}{ccc}
\underline{\text{Prover } (x, \chi, \psi)} & & \underline{\text{Verifier } (x)} \\
u_0, u_1 \leftarrow \mathbb{Z}_q & & \\
T := g_1^{u_0} g_2^{u_1} & \xrightarrow{\quad T \quad} & \\
 & \xleftarrow{\qquad\quad} & z \leftarrow \mathbb{Z}_q \\
s_0 := \chi \cdot z + u_0 & z & \\
s_1 := \psi \cdot z + u_1 & \xrightarrow{\quad s_0, s_1 \quad} & \\
 & & g_1^{s_0} g_2^{s_1} \overset{?}{=} T \cdot h^z \\
 & & c_1^{s_0} c_2^{s_1} \overset{?}{=} T \cdot (\frac{c_3}{\hat{\mathsf{pk}}})^z
\end{array}$$

Fig. 10. Σ-Protocol for Proof of Decryption

6 Conclusion

In this paper, we formalized the novel notion of signature schemes that are unforgeable under re-randomized keys. Furthermore, we showed that Schnorr's signature scheme [Sch90, Sch91] is unforgeable under re-randomized keys in the random oracle model and that Hofheinz' and Kiltz' signature scheme [HK08, HK12] is unforgeable under re-randomized keys in the standard model.

Based on signature schemes with re-randomizable keys we then gave a construction of unlinkable sanitizable signatures and an instantiation, which is at least one order of magnitude faster than all previously known schemes.

Acknowledgments

This work was supported by the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA – `www.cispa-security.org`) and the project PROMISE. Moreover, it was supported by the Initiative for Excellence of the German federal and state governments through funding for the Saarbrücken Graduate School of Computer Science and the DFG MMCI Cluster of Excellence. Part of this work was also supported by the German research foundation (DFG) through funding for the collaborative research center 1223. Dominique Schröder was also supported by an Intel Early Career Faculty Honor Program Award.

References

ACdT05. Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. Sanitizable signatures. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *ESORICS 2005: 10th European Symposium on Research in Computer Security*, volume 3679 of *Lecture Notes in Computer Science*, pages 159–177, Milan, Italy, September 12–14, 2005. Springer, Heidelberg, Germany.

ALP13. Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. Efficient completely context-hiding quotable and linearly homomorphic signatures. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 386–404, Nara, Japan, February 26 – March 1, 2013. Springer, Heidelberg, Germany.

BB04. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

BB08. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

BBD$^+$10. Christina Brzuska, Heike Busch, Özgür Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, and Dominique Schröder. Redactable signatures for tree-structured data: Definitions and constructions. In Jianying Zhou and Moti Yung, editors, *ACNS 10: 8th International Conference on Applied Cryptography and Network Security*, volume 6123 of *Lecture Notes in Computer Science*, pages 87–104, Beijing, China, June 22–25, 2010. Springer, Heidelberg, Germany.

BF11. Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 149–168, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

BF14. Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 520–537, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany.

BFF$^+$09. Christina Brzuska, Marc Fischlin, Tobias Freudenreich, Anja Lehmann, Marcus Page, Jakob Schelbert, Dominique Schröder, and Florian Volk. Security of sanitizable signatures revisited. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 317–336, Irvine, CA, USA, March 18–20, 2009. Springer, Heidelberg, Germany.

BFLS10. Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder. Unlinkability of sanitizable signatures. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 444–461, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany.

BGI14. Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 501–519, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany.

BMW03. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.

BPS12. Christina Brzuska, Henrich C. Pöhls, and Kai Samelin. Non-interactive public accountability for sanitizable signatures. In Sabrina De Capitani di Vimercati and Chris Mitchell, editors, *EuroPKI 2012: 9th European Workshop on Public Key Infrastructures, Services and Applications*, volume 7868 of *Lecture Notes in Computer Science*, pages 178–193, Pisa, Italy, September 13–14 2012. Springer, Heidelberg, Germany.

BPS13. Christina Brzuska, Henrich C. Pöhls, and Kai Samelin. Efficient and perfectly unlinkable sanitizable signatures without group signatures. In Sokratis Katsikas and Isaac Agudo, editors, *EuroPKI 2013: 10th European Workshop on Public Key Infrastructures, Services and Applications*, volume 8341 of *Lecture Notes in Computer Science*, pages 12–30, Egham, UK, September 12–13 2013. Springer, Heidelberg, Germany.

BPW03. Alexandra Boldyreva, Adriana Palacio, and Bogdan Warinschi. Secure proxy signature schemes for delegation of signing rights. Cryptology ePrint Archive, Report 2003/096, 2003. http://eprint.iacr.org/2003/096.

Cat14. Dario Catalano. Homomorphic signatures and message authentication codes. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14: 9th International Conference on Security in Communication Networks*, volume 8642 of *Lecture Notes in Computer Science*, pages 514–519, Amalfi, Italy, September 3–5, 2014. Springer, Heidelberg, Germany.

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, Santa Barbara, CA, USA, August 21–25, 1994. Springer, Heidelberg, Germany.

CJ10. Sébastien Canard and Amandine Jambert. On extended sanitizable signature schemes. In Josef Pieprzyk, editor, *Topics in Cryptology – CT-RSA 2010*, volume 5985 of *Lecture Notes in Computer Science*, pages 179–194, San Francisco, CA, USA, March 1–5, 2010. Springer, Heidelberg, Germany.

CJL12. Sébastien Canard, Amandine Jambert, and Roch Lescuyer. Sanitizable signatures with several signers and sanitizers. In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *AFRICACRYPT 12: 5th International Conference on Cryptology in Africa*, volume 7374 of *Lecture Notes in Computer Science*, pages 35–52, Ifrance, Morocco, July 10–12, 2012. Springer, Heidelberg, Germany.

CL04. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany.

CLX09. Ee-Chien Chang, Chee Liang Lim, and Jia Xu. Short redactable signatures using random trees. In Marc Fischlin, editor, *Topics in Cryptology – CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 133–147, San Francisco, CA, USA, April 20–24, 2009. Springer, Heidelberg, Germany.

CP93. David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Heidelberg, Germany.

CS98. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany.

DS15. David Derler and Daniel Slamanig. Rethinking privacy for extended sanitizable signatures and a black-box construction of strongly private schemes. In Man-Ho Au and Atsuko Miyaji, editors, *ProvSec 2015: 9th International Conference on Provable Security*, Lecture Notes in Computer Science, Kanazawa, Japan, November 24–26 2015. Springer, Heidelberg, Germany.

FF13. Marc Fischlin and Nils Fleischhacker. Limitations of the meta-reduction technique: The case of Schnorr signatures. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 444–460, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

Fra15. Pedro Franco. *Understanding Bitcoin: Cryptography, Engineering and Economics*. John Wiley & Sons, Chichester, UK, 2015.

Fre12. David Mandell Freeman. Improved security for linearly homomorphic signatures: A generic framework. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012: 15th International Conference on Theory and Practice of Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 697–714, Darmstadt, Germany, May 21–23, 2012. Springer, Heidelberg, Germany.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany.

FY05. Jun Furukawa and Shoko Yonezawa. Group signatures with separate and distributed authorities. In Carlo Blundo and Stelvio Cimato, editors, *SCN 04: 4th International Conference on Security in Communication Networks*, volume 3352 of *Lecture Notes in Computer Science*, pages 77–90, Amalfi, Italy, September 8–10, 2005. Springer, Heidelberg, Germany.

Gro07. Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180, Kuching, Malaysia, December 2–6, 2007. Springer, Heidelberg, Germany.

HK08. Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 21–38, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

HK12. Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. *Journal of Cryptology*, 25(3):484–527, July 2012.

JMSW02. Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In Bart Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 244–262, San Jose, CA, USA, February 18–22, 2002. Springer, Heidelberg, Germany.

JWL12. Rob Johnson, Leif Walsh, and Michael Lamb. Homomorphic signatures for digital photographs. In George Danezis, editor, *FC 2011: 15th International Conference on Financial Cryptography and Data Security*, volume 7035 of *Lecture Notes in Computer Science*, pages 141–157, Gros Islet, St. Lucia, February 28 – March 4, 2012. Springer, Heidelberg, Germany.

KL06. Marek Klonowski and Anna Lauks. Extended sanitizable signatures. In Min Surp Rhee and Byoungcheon Lee, editors, *ICISC 06: 9th International Conference on Information Security and Cryptology*, volume 4296 of *Lecture Notes in Computer Science*, pages 343–355, Busan, Korea, November 30 – December 1, 2006. Springer, Heidelberg, Germany.

MSK02. Shigeo Mitsunari, Ryuichi Saka, and Masao Kasahara. A new traitor tracing. *IEICE Transactions*, E85-A(2):481–484, February 2002.

PS96. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398, Saragossa, Spain, May 12–16, 1996. Springer, Heidelberg, Germany.

PS00. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

PS14. Henrich Christopher Pöhls and Kai Samelin. On updatable redactable signatures. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *ACNS 14: 12th International Conference on Applied Cryptography and Network Security*, volume 8479 of *Lecture Notes in Computer Science*, pages 457–475, Lausanne, Switzerland, June 10–13, 2014. Springer, Heidelberg, Germany.

SBZ02. Ron Steinfeld, Laurence Bull, and Yuliang Zheng. Content extraction signatures. In Kwangjo Kim, editor, *ICISC 01: 4th International Conference on Information Security and Cryptology*, volume 2288 of *Lecture Notes in Computer Science*, pages 285–304, Seoul, Korea, December 6–7, 2002. Springer, Heidelberg, Germany.

Sch90. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.

Sch91. Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.