Efficient Unlinkable Sanitizable Signatures from Signatures with Rerandomizable Keys

(Full Version)

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

Saarland University
fleischhacker@cs.uni-saarland.de, krupp@ca.cs.uni-saarland.de,
malavolta@cs.uni-saarland.de, s9joscne@stud.uni-saarland.de,
{schroeder,simkin}@ca.cs.uni-saarland.de

Abstract. Sanitizable signature schemes are a type of malleable signatures where the signer grants a designated third party, called the sanitizer, signing rights in the sense that the sanitizer can modify designated parts 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 one cannot link sanitized message-signature pairs of the same document and gave a generic construction based on group signatures that have a certain structure. Here, we present the first efficient instantiation of unlinkable sanitizable signatures. Our construction is based on a novel type of signature schemes with rerandomizable keys. Intuitively, this property allows to rerandomize both the signing and the verification key independently but consistently. This allows us to sign the message with a rerandomized 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 the fastest known construction.

1 Introduction

Sanitizable signature schemes were introduced by Ateniese et al. [1] and similar primitives were concurrently proposed by Steinfeld, Bull, and Zheng [26], by Miyazaki et al. [21], and by Johnson et al. [20]. 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 [1]. After the first introduction of sanitizable signatures in [1], the desired security properties were later formalized by Brzuska et al. [4]. At PKC 2010, Brzuska et al. [5] identified an important missing property called unlinkability. Loosely speaking, this notion says 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 a certain class of group signatures [2] in a blackbox-fashion. However, to this date, such generic constructions of unlinkable sanitizable signatures are inherently slow, since no efficient group signature scheme that has the required properties 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 [1,4] or we have to rely on an inefficient blackbox 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. 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.

	KGen_{sig}	KGen_{san}	Sign	Sanit	Verify	Proof	Judge
This paper	7 E	1 E	15 E	14 E	17 E	23 E	6 E
[5] using [17]	1 E	1 E	194 E + 2P	186 E + 1P	207 E + 62P	14 E + 1P	1 E + 2 P
[5] using [16]	1 E	4 E	2831 E	2814 E	2011 E	18 E	2 E

Table 1. Comparison of the dominant operations in our construction instantiated as described in Section 5 with the construction of Brzuska et al. [5] instantiated with Schnorr signatures and the group signature schemes of Groth [17] and Furukawa and Yonezawa [16] respectively. E and P stand for group exponentiations and pairing evaluations respectively.

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 rerandomizable keys*. This type of signature scheme allows to rerandomize both the signing and the verification key independently. It is required that the rerandomization is perfect, meaning that rerandomized 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 rerandomized key, even if the randomness is controlled by the attacker.

We show that this security notion is fulfilled by Schnorr's signature scheme [24,25], which is one of the most efficient signature schemes based on the discrete logarithm assumption. We also show that our notion is achievable in the standard model by slightly modifying the signature scheme of Hofheinz and Kiltz [18,19]. Apart from their usefulness in constructing highly efficient sanitizable signatures, this primitive might also be of independent interest.

Construction of Unlinkable Sanitizable Signature Schemes. Our construction of an unlinkable sanitizable signature scheme is based on signature schemes having perfectly rerandomizable keys. The main idea of the scheme is to sign the message with a rerandomized key and then prove, in zero-knowledge, that the key was derived from either the signer or the sanitizer. More precisely, to sign a message m, the signer extracts those parts of the message that cannot be modified by the sanitizer. We denote by m_{FIX} this fixed part of the message together with some auxiliary information such as the sanitizer's key. The signer then signs m_{FIX} with a (regular) strongly unforgeable signature scheme and signs the whole message together with both signer and sanitizer keys with a signature scheme that has perfectly rerandomizable keys. However, the signer cannot sign this part directly as this would reveal the identity of the signer. Instead, the signer chooses a fresh randomness ρ and rerandomizes his key-pair and appends a zero-knowledge proof that the derived keys descend either from signer's or the sanitizer's key. Sanitizing a message follows the same idea: the sanitizer modifies the message and signs it with a rerandomized version of his key pair and appends a zero-knowledge proof for the same language. To turn this idea into an efficient scheme, we do not use (inefficient) generic instantiations of zero-knowledge, but an efficient sigma protocol tailored to our problem that we then convert via the Fiat-Shamir transformation [14] into an efficient non-interactive zero-knowledge proof.

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 suitably instantiated versions of the scheme due to Brzuska et al. [5]. We instantiate the signature scheme in [5] using a deterministic version of Schnorr's signature scheme. Instantiating the group signature scheme used in [5] on the other hand is quite hard. The construction requires very specific properties of the group signature scheme. Namely, it is required that user keys can be generated independently of and, in particular, before the group manager's key. This property originates from the definitions of Bellare, Micciancio, and Warinschi [2], however almost no efficient group signature scheme follows their definitions. Only very few group signature schemes such as [17,16] can be adapted to have this property and at the same time fulfill all security requirements needed in [5].

We instantiate [5] with the group signature schemes of Groth [17] and of Furukawa and Yonezawa [16]. To the best of our knowledge, these are the two most efficient group signature schemes that can be adapted to allow an instantiation of [5].

In Table 1 we provide a comprehensive comparison of the computational costs of all algorithms. It is easy to see that in the most important algorithms, i.e., signing, sanitizing, and verification, our construction is at least one order of

[2] 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 [5] using [17], since group signatures in this scheme consist exclusively of $\mathbb{G}$-elements.

	pk_{sig}	sk_{sig}	pk_{san}	sk_{san}	σ	π
This paper	7	14	1	1	14	4
[5] using [17]	1	1	1	1	69	1
[5] using [16]	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. [5] instantiated with Schnorr signatures and the group signature schemes of Groth [17] and Furukawa and Yonezawa [16] 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.[2]

magnitude faster than both instantiations of [5]. Similarly, Table 2 provides an overview of the storage complexity of the different constructions. It is easy to see that the signatures in our construction are the smallest by a large margin.

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.

1.3 Related Work

Ateniese et al. [1] 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. [4] and the *unlinkability* property was introduced by Brzuska et al. in [5]. Later, in [6], 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 [7], 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 by Sebastien Canard and Amandine Jambert [9] 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 [10], Canard et al. extend sanitizable signatures to the setting with multiple signers and sanitizers.

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 $\textsc{Adm}$ for this particular message and sanitizer. The sanitizer may then change the message according to some modification $\textsc{Mod}$ and update the signature using his 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 $\textsc{Adm}$ 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 $\textsc{Mod}$ 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 [4,5] 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

In this section we recall the definitions of sanitizable signature schemes [4,5].

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:*

$(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\kappa)$*: The signer key generation algorithm takes as input the security parameter* 1^κ *and generates a key pair* $(\mathsf{sk}_{sig}, \mathsf{pk}_{sig})$*.*

$(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{KGen}_{san}(1^\kappa)$*: The sanitizer key generation algorithm takes as input the security parameter* 1^κ *and generates a key pair* $(\mathsf{sk}_{san}, \mathsf{pk}_{san})$*.*

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

$\{(m', \sigma'), \bot\} \leftarrow \mathsf{Sanit}(m, \textsc{Mod}, \sigma, \mathsf{pk}_{sig}, \mathsf{sk}_{san})$*: 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* $\bot$ *in case of an error.*

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

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

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

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 [4].

2.2 Security of Sanitizable Signatures

In this section we recall the definitions of the necessary security properties of immutability, accountability, transparency, and unlinkability from [5]. Due to space constraints, we keep the descriptions of the definitions short and omit most definitions of the oracles (unless the functionality is not obvious); we refer the reader to [4,5] for comprehensive descriptions and discussions. Weaker security notions, such as unforgeability and privacy are already implied by accountability and transparency, respectively, as shown in [4].

Immutability. Informally, this property says that a malicious sanitizer cannot change inadmissible blocks.

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

```
Experiment $\mathsf{Immut}_{\mathcal{A}}^{\mathrm{SanS}}(\kappa)$
    $(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\kappa)$
    $(\mathsf{pk}^*_{san}, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, \mathsf{sk}_{sig}, \cdot, \cdot), \mathsf{Proof}(\mathsf{sk}_{sig}, \cdot, \cdot, \cdot)}(\mathsf{pk}_{sig})$
        letting $(m_i, \textsc{Adm}_i, \mathsf{pk}_{san,i})$ and $\sigma_i$ denote the
```

```
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 $(m_i, \mathrm{ADM}_i, \mathsf{pk}_{san,i})$ 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
        $\mathsf{Judge}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san}, \pi) \neq \mathtt{San}$
```

```
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 $(m_i, \mathrm{MOD}_i, \sigma_i, \mathsf{pk}_{sig,i})$ 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
        $\mathsf{Judge}(m^*, \sigma^*, \mathsf{pk}^*_{sig}, \mathsf{pk}_{san}, \pi^*) \neq \mathtt{Sign}$
    else output 0.
```

Fig. 1. Sanitizer- and Signer-accountability experiments.

```
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 $\left(a = b \text{ and } M_{\mathsf{Sanit/Sign}} \cap M_{\mathsf{Proof}} = \emptyset\right)$
    Else output 0

The oracle Sanit/Sign is defined as:
Sanit/Sign$(m, \mathrm{MOD}, \mathrm{ADM})$
    $\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}_{sig}, \mathsf{pk}_{san}, \mathrm{ADM})$
    $(m', \sigma_0) \leftarrow \mathsf{Sanit}(m, \mathrm{MOD}, \sigma, \mathsf{pk}_{sig}, \mathsf{sk}_{san})$
    $\sigma_1 \leftarrow \mathsf{Sign}(m', \mathsf{sk}_{sig}, \mathsf{pk}_{san}, \mathrm{ADM})$
    output $(m', \sigma_b)$
```

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

The oracle LoRSanit is defined as:
LoRSanit$((m_0, \mathrm{MOD}_0, \sigma_0), (m_1, \mathrm{MOD}_1, \sigma_1))$
    if $\mathsf{Verify}(m_i, \sigma_i, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 1$
      and $\mathrm{ADM}_i(\mathrm{MOD}_i) \neq 0$ for $i \in \{0,1\}$, and
      $\mathrm{MOD}_0(m_0) = \mathrm{MOD}_1(m_1)$ and $\mathrm{ADM}_0 = \mathrm{ADM}_1$
    then
      $(m', \sigma') \leftarrow \mathsf{Sanit}(m_b, \mathrm{MOD}_b, \sigma_b, \mathsf{pk}_{sig}, \mathsf{sk}_{san})$
      output $(m', \sigma')$
    else output $\perp$.
```

Fig. 2. The experiments for transparency and unlinkability.

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:*

$$\mathsf{pk}^*_{san} \neq \mathsf{pk}_{san,i} \text{ or } m^* \notin \{\mathrm{MOD}(m_i) \mid \mathrm{MOD} \text{ with } \mathrm{ADM}_i(\mathrm{MOD}) = 1\}$$

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.

Definition 3 (Sanitizer-Accountability). *A sanitizable signature scheme* SanS *is called* 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 the experiment $\mathsf{San\text{-}Acc}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$ is defined in Figure 1.*

Definition 4 (Signer-Accountability). *A sanitizable signature scheme* SanS *is called* 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 the experiment $\mathsf{Sig\text{-}Acc}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$ is defined in Figure 1.*

Transparency. Informally, this property says that one cannot decide whether a signature has been sanitized or not.

Definition 5 (Proof-Restricted Transparency). *A sanitizable signature scheme* SanS *is* 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 the experiment $\mathsf{Trans}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$ is defined in Figure 2.*

Unlinkability. This security notion demands that it is not possible to use the signatures to identify sanitized message-signature pairs originating from the same source.

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

3 Unforgeable Signatures Schemes Under Rerandomizable Keys

Our sanitizable signature scheme is based on signatures that are unforgeable under rerandomizable keys, meaning that it is computationally hard to forge a signature even if the adversary learns signatures on messages under rerandomized keys (where the randomness to rerandomize the keys is chosen by the attacker). This new notion for digital signatures may also be of independent interest. To define this property and the corresponding security notion formally, we denote by $pp \leftarrow \mathsf{SSetup}(1^\kappa), (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa), \sigma \leftarrow \mathsf{SSign}(\mathsf{sk}, m), b \leftarrow \mathsf{SVerify}(\mathsf{pk}, m, \sigma)$ the standard algorithms of a digital signature scheme.

Definition 7 (Digital Signature Scheme with Perfectly Rerandomizable Keys). *A signature scheme* $\Sigma = (\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify})$ *has perfectly rerandomizable keys if there exist two additional PPT algorithms* $(\mathsf{RandSK}, \mathsf{RandPK})$ *and a randomness space* χ *such that:*

$\mathsf{RandSK}(\mathsf{sk}, \rho)$*: The secret key rerandomization 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 rerandomization 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* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$*, all* $m \in \{0,1\}^*$*, and* $\sigma \leftarrow \mathsf{SSign}(\mathsf{sk}, m)$*, it holds that* $\mathsf{SVerify}(\mathsf{pk}, m, \sigma) = 1$.
2. *For all* $\kappa \in \mathbb{N}$*, all* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$*, all randomness* ρ*, all* $m \in \{0,1\}^*$*, 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* ρ*, 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* $(\mathsf{sk}'', \mathsf{pk}'') \leftarrow \mathsf{SGen}(1^\kappa)$

In the following, we define unforgeability under rerandomized key attacks. 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 rerandomizes the private key according to ρ and signs the message using this key.

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

```
***Experiment*** $\mathsf{UFRK}_{\mathcal{A}}^{\Sigma}(\kappa)$ :
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$
    $Q := \emptyset$
    $(m^*, \sigma^*, \rho^*) \leftarrow \mathcal{A}^{\mathcal{O}_1(\mathsf{sk}, \cdot), \mathcal{O}_2(\mathsf{sk}, \cdot, \cdot)}(\mathsf{pk})$
    *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 rerandomizable 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* $\mathsf{EUF}_{\mathcal{A}}^{\Sigma}(\kappa)$ *evaluates to 1 is negligible (in* κ*), where* $\mathsf{EUF}_{\mathcal{A}}^{\Sigma}(\kappa)$ *is defined as* $\mathsf{UFRK}_{\mathcal{A}}^{\Sigma}(\kappa)$*, but the adversary only gets access to* $\mathcal{O}_1$ *and wins if the first condition is fulfilled.*

Observe that Definition 8 does not trivially follow from Definition 9. In fact very few standard model signatures can be proven secure according to definition Definition 8 and some well known schemes are trivially forgeable under rerandomized keys. Examples include the signature schemes due to Boneh and Boyen [3] and Camenisch and Lysyanskaya [8]. The attacks to these schemes are sketched in Appendix A.

Definition 10 (Strong Existential Unforgeability). *A signature scheme with perfectly rerandomizable keys* $\Sigma = (\mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify}, \mathsf{RandSK}, \mathsf{RandPK})$ *is* strongly existentially unforgeable under chosen message attacks (s-EUF) *if for all PPT adversaries* $\mathcal{A}$ *the probability that the experiment* $\mathsf{s\text{-}EUF}_{\mathcal{A}}^{\Sigma}(\kappa)$ *evaluates to 1 is negligible (in* κ*), where* $\mathsf{s\text{-}EUF}_{\mathcal{A}}^{\Sigma}(\kappa)$ *is defined as* $\mathsf{UFRK}_{\mathcal{A}}^{\Sigma}(\kappa)$*, 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* $(m^*, \sigma^*) \notin Q$*.*

3.1 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 [24,25] is unforgeable under rerandomized keys and in the standard model we show that a slightly modified version of the signature scheme due to Hofheinz and Kiltz [18,19] satisfies our notion.

Random Oracle Model We show that Schnorr's signature scheme [24,25] is unforgeable under rerandomized keys. Our proof technique is based on a trick – first observed by Fischlin and Fleischhacker [15] in the context of an impossibility result – that allows moving a signature from one public key to another 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 rerandomized keys.

Theorem 1 (Unforgeability of Schnorr Signatures Under Rerandomized Keys). *The signature scheme* SSS *as defined in Definition 11 is unforgeable under rerandomized keys as defined in 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 adversary $\mathcal{A}$ such that the probability that the experiment $\mathsf{UFRK}_{\mathcal{A}}^{\mathsf{SSS}}$ evaluates to 1 is at least $1/\mathsf{poly}(\kappa)$. We then construct an adversary $\mathcal{B}$ against the existential unforgeability of SSS defined as follows:

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

where $\mathcal{B}$ answers all queries to $\mathcal{O}_1(\mathsf{sk}, m)$ with its own oracle and the simulation of $\mathcal{O}_2(\mathsf{sk}, m)$ is shown on the right.

It is easy to see that the simulation of $\mathcal{A}$'s signing oracle is perfect. 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 $\mathsf{UFRK}^{\mathsf{SSS}}_{\mathcal{A}}(\kappa)$ 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 $\mathsf{UFRK}^{\mathsf{SSS}}_{\mathcal{A}}(\kappa)$ experiment, $\mathcal{B}$ wins in the $\mathsf{EUF}^{\mathsf{SSS}}_{\mathcal{A}}(\kappa)$ experiment. Combining this with the well known proof of existential unforgeability of Schnorr signatures by Pointcheval and Stern [22,23] rules out the existence of $\mathcal{A}$ under the discrete logarithm assumption in the random oracle model.

Standard Model In the standard model an instantiation of a signature scheme with unforgeability under rerandomized keys can be found in the work of Hofheinz and Kiltz [18,19]. To prove this formally, we adapt the technique that we used in the proof of Schnorr's signature scheme to this setting, which allows us to reduce the unforgeability under rerandomized keys to standard existential unforgeability.

The scheme of Hofheinz and Kiltz requires a programmable hash function [18,19], but since security properties of programmable hash functions are not relevant to our proofs, we therefore omit them here and refer the interested reader to [18,19]. It should be noted that we slightly adapt the Hofheinz Kiltz signature scheme, which works on type 1 and type 2 pairings. In particular, in SSign we choose s as a random element from $\mathbb{Z}_q$ and not as a random bit string. This slightly increases the signature's size, but does not influence the original functionality or security proof.

Definition 12 (Programmable Hash Function [18,19]). *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 review the signature scheme due to Hofheinz Kiltz and define the rerandomization algorithms.

Definition 13 (Hofheinz Kiltz Signature Scheme [18,19]). *Let* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ *be a bilinear map. Let* g_1 *and* g_2 *be generators of* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*, respectively. 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* $pp = k$*.*

$\mathsf{SGen}(1^\kappa)$*: 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 rerandomized keys.

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

Proof. Assume towards contradiction that there exists an adversary $\mathcal{A}$ such that the probability that the experiment $\mathsf{UFRK}_{\mathcal{A}}^{\mathsf{HKSS}}(\kappa)$ evaluates to is bigger than $1/\mathsf{poly}(\kappa)$.

We then construct an adversary $\mathcal{B}$ against the existential unforgeability of HKSS as follows:

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

where $\mathcal{B}$ answers all queries to $\mathcal{O}_1(\mathsf{sk}, m)$ with its own oracle and the simulation of $\mathcal{O}_2(\mathsf{sk}, m)$ is shown on the right. It is easy to see that $\mathcal{A}$'s simulation of the signing oracle is perfect, since whenever an adversary that sends (ρ, m) and receives a signature (s', y) then it holds that $e(y, \mathsf{pk} \cdot g_2^{\rho} \cdot g_2^{s'}) = e(\mathsf{Eval}(k, m)^{\frac{1}{\mathsf{sk}+s}}, g_2^{\mathsf{sk}+\rho+(s-\rho)}) = e(\mathsf{Eval}(k, m), g_2)$.

Similarly, $\mathcal{B}$ outputs a valid signature $(s' = s + \rho^*, y)$ for m^* under pk, whenever $\mathcal{A}$ returns a valid signature (s, y) for m^* under the rerandomized 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 Hofheinz Kiltz signatures from [18,19] 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 rerandomizable keys. After defining the required preliminaries in Section 4.1, we provide a formal description of our construction in Section 4.2. We will provide a proof of security in Section 4.3. A concrete instantiation of our construction will be given in Section 5

4.1 Preliminaries

In this section we shortly 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*.*

<u>CORRECTNESS:</u> *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* $(\mathsf{S_{ZK}}, \mathsf{P_{ZK}}, \mathsf{V_{ZK}})$ *for a language* $\mathcal{L}$ *with the corresponding relation* $\mathcal{R}$ *consists of three algorithms:*

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

$\mathsf{P_{ZK}}(crs, x, w)$*: 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*.*

$\mathsf{V_{ZK}}(crs, x, \pi)$*: 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*

$$\Pr[crs \leftarrow \mathsf{S_{ZK}}(1^\kappa); (x, w) \leftarrow \mathcal{A}(crs); \pi \leftarrow \mathsf{P_{ZK}}(crs, x, w); \mathsf{V_{ZK}}(crs, x, \pi) = 1 \mid x \in \mathcal{L}] = 1$$

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*

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

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*

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

where $\mathsf{S}'(crs, \mathrm{T}, x, w) = \mathsf{S}_1(crs, \mathrm{T}, x)$ *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

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

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

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

4.2 Our Construction

In the following, we describe our construction of a sanitizable signature scheme based on signatures with rerandomizable keys. The basic idea is that a signature of our scheme is comprised of a public key pk', a signature of the message, which is valid under this key pk', and a zero-knowledge proof that pk' is a rerandomization of the signer's or the sanitizer's public key. To allow for an easy Proof and Judge algorithm, we have to provide a way to check that pk' is in fact the rerandomization of the signer's or the sanitizer's public key. Therefore, we also include an encryption of the actual public key, so that the signer can provably decrypt. Similar to previous constructions [4,5], our signature also contains a second signature, signed by the signer, on the fixed part of the message and a description of valid modifications ADM. We require a signature scheme with perfectly rerandomizable keys $\Sigma = (\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify}, \mathsf{RandSK}, \mathsf{RandPK})$ that is unforgeable under rerandomized keys, a deterministic strongly existentially unforgeable signature scheme $\Sigma_{\mathrm{FIX}} = (\mathsf{SSetup}_{\mathrm{FIX}}, \mathsf{SGen}_{\mathrm{FIX}}, \mathsf{SSign}_{\mathrm{FIX}}, \mathsf{SVerify}_{\mathrm{FIX}})$, a CCA secure public key encryption scheme $\mathcal{E} = (\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec})$ as well as two perfectly sound non-interactive zero-knowledge proof systems $\Pi_{PoK} = (\mathsf{S}_{\mathsf{PoK}}, \mathsf{P}_{\mathsf{PoK}}, \mathsf{V}_{\mathsf{PoK}})$ and $\Pi_{ZK} = (\mathsf{S}_{\mathsf{ZK}}, \mathsf{P}_{\mathsf{ZK}}, \mathsf{V}_{\mathsf{ZK}})$ for the two languages $\mathcal{L}_1$ and $\mathcal{L}_2$ specified below. The proof system Π_{PoK} is also required to be a proof of knowledge.

The two languages of our the proof systems are defined as follows: Intuitively, the first one says that the ciphertext c contains the encryption of the signer's (resp. sanitizer's) public key and the randomness that was used to derive the rerandomized key pk'. More formally, the language $\mathcal{L}_1$ 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 $\mathcal{L}_2$ is used for the proof algorithm and contains tuples $(\mathsf{ek}, c, \widehat{\mathsf{pk}})$ for which there exists witness $w = (\psi, \mathsf{dk})$ such that

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

Given the primitives specified above and the definitions of $\mathcal{L}_1$ and $\mathcal{L}_2$, we are ready to define our sanitizable signature scheme $\mathrm{SanS} = (\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$ 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:

```
$\underline{\mathsf{Setup}(1^\kappa):}$
    $crs_{PoK} \leftarrow \mathsf{S}_{\mathsf{PoK}}(1^\kappa)$
    $crs_{ZK} \leftarrow \mathsf{S}_{\mathsf{ZK}}(1^\kappa)$
    $pp \leftarrow \mathsf{SSetup}(1^\kappa)$

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

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

Signing and Sanitizing. The signing and sanitizing are very similar algorithms. In both cases the algorithm first parse their inputs and Sanit further checks that $\textsc{Mod}$ 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 rerandomizing the respective key, encrypting the original key, proving that sk' is indeed a rerandomization and signing the full message together with signer's and sanitizer's public keys as seen in the following:

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

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

Verification. The verification of a signature is rather simple. The verification algorithm simply 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 $(\mathsf{pk}_{\mathrm{FIX}}, \mathsf{pk}, \mathsf{ek})$.
    Parse $\sigma$ as $(\sigma_{\mathrm{FIX}}, \sigma', \textsc{Adm}, \mathsf{pk}', c, \tau)$.
    $m_{\mathrm{FIX}} := (\textsc{Fix}_{\textsc{Adm}}(m), \textsc{Adm}, \mathsf{pk}_{san})$
    $\mathsf{stmt} := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}')$
    if $\left(\begin{array}{rl} & \mathsf{SVerify}_{\mathrm{FIX}}(\mathsf{pk}_{\mathrm{FIX}}, m_{\mathrm{FIX}}, \sigma_{\mathrm{FIX}}) = 1 \\ \text{and} & \mathsf{SVerify}(\mathsf{pk}', (m, \mathsf{pk}_{sig}, \mathsf{pk}_{san}), \sigma') = 1 \\ \text{and} & \mathsf{V}_{\mathsf{PoK}}(crs, \mathsf{stmt}, \tau) = 1 \end{array}\right)$
    then output 1
    else output 0
```

Proving and Judging. To prove who computed an accused signature, the signer 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. To convince the judge, the signer further computes a zero knowledge proof asserting that the decryption was performed correctly. The Judge algorithm therefore, naturally, checks whether the proof of decryption is correct. If the proof π further contains pk_{san}, Judge decides that the signature was indeed produced by the sanitizer. In all other cases, Judge defaults to blaming the signer. This follows naturally from the fact that the signer also serves as the authority in a sanitizable signature scheme and is thus much more powerful.

```
$\underline{\mathsf{Proof}(\mathsf{sk}_{sig}, m, \sigma, \mathsf{pk}_{san}):}$
  If $\mathsf{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 0$
    output $\bot$
  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 $(\sigma_{\text{FIX}}, \sigma', \text{ADM}, \mathsf{pk}', c, \tau)$.
  $\widehat{\mathsf{pk}} \leftarrow \mathsf{Dec}(\mathsf{dk}, c)$
  $\mathsf{stmt} := (\mathsf{ek}, c, \widehat{\mathsf{pk}})$
  $\phi \leftarrow \mathsf{P}_{\mathsf{ZK}}(crs, \mathsf{stmt}, (\psi, \mathsf{dk}))$
  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 $(\widehat{\mathsf{pk}}, \phi)$.
  $\mathsf{stmt} := (\mathsf{ek}, c, \widehat{\mathsf{pk}})$
  if $\left(\begin{array}{c} \mathsf{pk}_{san} = \widehat{\mathsf{pk}} \\ \text{and } \mathsf{V}_{\mathsf{ZK}}(crs, \mathsf{stmt}, \phi) = 1 \end{array}\right)$
  then output $\mathtt{San}$
  else output $\mathtt{Sign}$
```

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 rerandomizable keys that is unforgeable under rerandomized keys $\Pi_{ZK} = (\mathsf{S}_{\mathsf{ZK}}, \mathsf{P}_{\mathsf{ZK}}, \mathsf{V}_{\mathsf{ZK}})$ is a perfectly sound non-interactive zero knowledge proof system, and $\Pi_{PoK} = (\mathsf{S}_{\mathsf{PoK}}, \mathsf{P}_{\mathsf{PoK}}, \mathsf{V}_{\mathsf{PoK}})$ is a perfectly 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 $\mathsf{San\text{-}Acc}^{\text{SanS}}_{\mathcal{A}}(\kappa) = 1$, 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 $(m_i, \text{ADM}_i, \mathsf{pk}_{san,i})$ denotes the ith query to the Sign oracle.

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

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

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

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

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

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

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

$\mathsf{Sign}'(m, \mathsf{pk}_{san}, \text{ADM})$ :
    $\rho \leftarrow \chi$
    $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}, \rho)$
    $c \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{pk}; \omega)$
    $\mathsf{stmt} := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}')$
    $\tau \leftarrow \mathsf{P}_{\mathsf{PoK}}(crs_{PoK}, \mathsf{stmt}, (\rho, \omega))$
    $m_{\text{FIX}} := (\text{FIX}_{\text{ADM}}(m), \text{ADM}, \mathsf{pk}_{san})$
    $\sigma_{\text{FIX}} \leftarrow \mathsf{SSign}_{\text{FIX}}(\mathsf{sk}_{\text{FIX}}, m_{\text{FIX}})$
    $\sigma' \leftarrow \mathcal{O}_2(\mathsf{sk}, (m, \mathsf{pk}_{sig}, \mathsf{pk}_{san}), \rho)$
    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 $\perp$
    $\widehat{\mathsf{pk}} \leftarrow \mathsf{Dec}(\mathsf{dk}, c)$
    $\mathsf{stmt} := (\mathsf{ek}, c, \widehat{\mathsf{pk}})$
    $\phi \leftarrow \mathsf{P}_{\mathsf{ZK}}(crs_{ZK}, \mathsf{stmt}, (\psi, \mathsf{dk}))$
    output $(\widehat{\mathsf{pk}}, \phi)$
```

Fig. 3. Description of reduction $\mathcal{B}_1$, reducing the sanitizer accountability of SanS against the UFRK security of Σ.

Now, consider reduction $\mathcal{B}_1$, depicted in Figure 3 against the unforgeability under rerandomized keys of the underlying signature scheme. Observe that this reduction is clearly efficient and perfectly simulates the view of $\mathcal{A}$ in the game $\mathsf{San\text{-}Acc}_{\mathcal{A}}^{\mathrm{SanS}}(\kappa)$. 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

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

which must be negligible because the signature scheme is unforgeable under rerandomized keys.

Thus it must hold that $\Pr\left[\mathsf{San\text{-}Acc}_{\mathcal{A}}^{\mathrm{SanS}}(\kappa) = 1\right]$ 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 rerandomizable keys that is unforgeable under rerandomized keys and $\Pi_{PoK} = (\mathsf{S}_{\mathsf{PoK}}, \mathsf{P}_{\mathsf{PoK}}, \mathsf{V}_{\mathsf{PoK}})$ is a perfectly non-interactive zero-knowledge proof of knowledge system, 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 $(\mathsf{pk}_{\text{FIX}}{}^*, \mathsf{pk}^*, \mathsf{ek}^*)$, σ^* can be parsed as $(\sigma_{\text{FIX}}, \sigma', \text{ADM}, \mathsf{pk}', c, \tau)$, and π^* can be parsed as $(\widehat{\mathsf{pk}}, \phi)$.

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

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

where $(m_i, \textsc{Mod}_i, \sigma_i, \mathsf{pk}_{sig,i})$ 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

$$\mathsf{SVerify}(\mathsf{pk}', (m^*, \mathsf{pk}_{sig}^*, \mathsf{pk}_{san}), \sigma') = 1 \tag{13}$$
$$\mathsf{V}_{\mathsf{PoK}}(crs_{PoK}, (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}'), \tau) = 1. \tag{14}$$

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

$$\mathsf{pk}_{san} = \widehat{\mathsf{pk}} \tag{15}$$
$$\mathsf{V}_{\mathsf{ZK}}(crs_{ZK}, (\mathsf{ek}^*, c, \widehat{\mathsf{pk}}), \phi) = 1. \tag{16}$$

Now, consider reduction $\mathcal{B}_2$, depicted in Figure 4 against the unforgeability under rerandomized keys of the underlying signature scheme.

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

$\underline{\mathsf{Sanit}'(m, \sigma, \textsc{Mod}, \mathsf{pk}_{sig}):}$
    Parse $\mathsf{pk}_{sig}$ as $(\mathsf{pk}_{\mathrm{FIX}}, \mathsf{pk}, \mathsf{ek})$.
    Parse $\sigma$ as $(\sigma_{\mathrm{FIX}}, \sigma', \textsc{Adm}, \mathsf{pk}', c, \tau)$.
    If $\textsc{Adm}(\textsc{Mod}) = 0$
        output $\perp$
    $\widehat{m}' := \textsc{Mod}(m)$
    $\rho \leftarrow \chi$
    $\widehat{\mathsf{pk}}' \leftarrow \mathsf{RandPK}(\mathsf{pk}_{san}, \rho)$
    $\widehat{c} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{pk}_{san}; \omega)$
    $\mathsf{stmt} := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \widehat{\mathsf{pk}}')$
    $\widehat{\tau} \leftarrow \mathsf{P}_{\mathsf{PoK}}(crs_{PoK}, \mathsf{stmt}, (\rho, \omega))$
    $\widehat{\sigma}' \leftarrow \mathcal{O}_2(\mathsf{sk}, (\widehat{m}', \mathsf{pk}_{sig}, \mathsf{pk}_{san}), \rho)$
    $\widehat{\sigma} = (\sigma_{\mathrm{FIX}}, \widehat{\sigma}', \textsc{Adm}, \widehat{\mathsf{pk}}', \widehat{c}, \widehat{\tau})$
    output $(m', \widehat{\sigma})$
```

Fig. 4. Description of reduction $\mathcal{B}_2$, reducing the signer accountability of SanS against the UFRK security of Σ.

Observe that this reduction is clearly efficient and perfectly simulates the view of $\mathcal{A}$ in the game $\mathsf{Sig\text{-}Acc}_{\mathcal{A}}^{\mathrm{SanS}}(\kappa)$. 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

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

which must be negligible because the signature scheme is unforgeable under rerandomized keys.

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

Theorem 5 (Immutability). *If* $\Sigma_{\text{FIX}} = (\mathsf{SSetup}_{\text{FIX}}, \mathsf{SGen}_{\text{FIX}}, \mathsf{SSign}_{\text{FIX}}, \mathsf{SVerify}_{\text{FIX}})$ *is a deterministic signature scheme that 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 $\mathsf{Immut}^{\text{SanS}}_{\mathcal{A}}(\kappa) = 1$, 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

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

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

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 5 against the strong existential unforgeability of the underlying signature scheme.

```
$\mathcal{B}_3^{\mathcal{O}(\mathsf{sk},\cdot)}(\mathsf{pk}_{\text{FIX}})$ :
  $crs_{PoK} \leftarrow \mathsf{S}_{\mathsf{PoK}}(1^\kappa)$
  $crs_{ZK} \leftarrow \mathsf{S}_{\mathsf{ZK}}(1^\kappa)$
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$
  $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{EGen}(1^\kappa; \psi)$
  $\mathsf{pk}_{sig} := (\mathsf{pk}_{\text{FIX}}, \mathsf{pk}, \mathsf{ek})$
  $(m^*, \sigma^*, \mathsf{pk}^*_{san}) \leftarrow \mathcal{A}^{\mathsf{Sign}'(\cdot,\cdot,\cdot), \mathsf{Proof}(\mathsf{sk}_{sig},\cdot,\cdot)}(\mathsf{pk}_{sig})$
  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 $(m^*_{\text{FIX}}, \sigma^*_{\text{FIX}})$
```

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

Fig. 5. 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 $\Pi_{PoK} = (\mathsf{S}_{\mathsf{PoK}}, \mathsf{P}_{\mathsf{PoK}}, \mathsf{V}_{\mathsf{PoK}})$ is a computationally zero-knowledge perfectly sound proof of knowledge system, $\Pi_{ZK} = (\mathsf{S}_{\mathsf{ZK}}, \mathsf{P}_{\mathsf{ZK}}, \mathsf{V}_{\mathsf{ZK}})$ 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 rerandomizable keys that is unforgeable under rerandomized keys, then* SanS *is (proof-restrictedly) transparent.*

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

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

Game_1 works exactly as Game_0, except that crs_{PoK} is chosen as $(crs_{PoK}, \mathrm{T}_{\mathrm{PoK}}) \leftarrow \mathsf{S}_{\mathsf{PoK},0}(1^\kappa)$ and the proofs τ in the answers to $\mathsf{Sanit}/\mathsf{Sign}$ queries are computed as $\tau \leftarrow \mathsf{S}_{\mathsf{PoK},1}(crs, \mathrm{T}, \mathsf{stmt})$, 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 crs_{ZK} is chosen as $(crs_{ZK}, \mathrm{T}_{\mathrm{ZK}}) \leftarrow \mathsf{S}_{\mathsf{ZK},0}(1^\kappa)$ and the proofs of decryption ϕ in the answers to Proof queries are computed as $\phi \leftarrow \mathsf{S}_{\mathsf{ZK},1}(crs, \mathrm{T}, (\psi, \mathsf{dk}))$, 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 $\mathsf{Sanit}/\mathsf{Sign}$ 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 crs_{ZK} is once again chosen honestly as $crs_{ZK} \leftarrow \mathsf{S}_{\mathsf{ZK}}(1^\kappa)$ and the proofs of decryption ϕ in the answers to Proof queries are computed honestly as $\phi \leftarrow \mathsf{P}_{\mathsf{ZK}}(crs_{ZK}, \mathsf{stmt}, (\psi, \mathsf{dk}))$.

Game_7 works exactly as Game_6, except that crs_{PoK} is once again chosen honestly as $crs_{PoK} \leftarrow \mathsf{S}_{\mathsf{ZK}}(1^\kappa)$ and the proofs τ in the answers to $\mathsf{Sanit}/\mathsf{Sign}$ queries are computed honestly as $\tau \leftarrow \mathsf{P}_{\mathsf{PoK}}(crs_{PoK}, \mathsf{stmt}, (\rho, \omega))$. 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 6 against the zero-knowledge property of the underlying proof of knowledge system.

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

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

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 7 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

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

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

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

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

Fig. 6. 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}.

$\mathsf{Game}_2 \approx \mathsf{Game}_3$ 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 8 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 $\mathsf{IND\text{-}CCA}$ 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.

$\mathsf{Game}_3 \approx \mathsf{Game}_4$ 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 rerandomized and in Game_4, the sanitizer's key is rerandomized. However, by virtue of the perfect rerandomizability property of the signature scheme, the rerandomized keys are in fact distributed identically in both cases. Further, the remainder of the signature is computed independently from the rerandomization 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 $(m, \sigma = (\sigma_{\mathrm{FIX}}, \sigma', \textsc{Adm}, \mathsf{pk}', c, \tau), \mathsf{pk}'_{san})$ such that the following conditions hold.

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

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

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

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

Fig. 7. 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}.

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

$\mathsf{Sanit/Sign}'(m, \mathrm{MOD}, \mathrm{ADM})$ :
  $\rho \leftarrow \chi$
  $\mathsf{sk}' \leftarrow \mathsf{RandSK}(\mathsf{sk}, \rho)$
  $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}, \rho)$
  $\sigma_{\mathrm{FIX}} := \mathsf{SSign}_{\mathrm{FIX}}(\mathsf{sk}_{\mathrm{FIX}}, m_{\mathrm{FIX}})$
  $m' := \mathrm{MOD}(m)$
  $\sigma' := \mathsf{SSign}(\mathsf{sk}', m')$
  $c \leftarrow \mathcal{O}(\mathsf{pk}, \overline{\mathsf{pk}})$
  $C_{\mathsf{Sanit/Sign}} := C_{\mathsf{Sanit/Sign}} \cup \{c\}$
  $\mathsf{stmt} := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}')$
  $\tau \leftarrow \mathsf{S}_{\mathsf{PoK},1}(crs_{PoK}, \mathrm{T}_{\mathrm{PoK}}, \mathsf{stmt})$
  $\sigma := (\sigma_{\mathrm{FIX}}, \sigma', \mathrm{ADM}, \mathsf{pk}', c, \tau)$
  return $(m', \sigma)$

$\mathsf{Proof}'(m, \sigma, \mathsf{pk}_{san})$ :
  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
    $\widehat{\mathsf{pk}} \leftarrow \mathsf{Dec}(c)$
  $\mathsf{stmt} := (\mathsf{ek}, c, \widehat{\mathsf{pk}})$
  $\phi \leftarrow \mathsf{S}_{\mathsf{ZK},(}crs_{ZK}, \mathsf{stmt})$
  output $(\widehat{\mathsf{pk}}, \phi)$
```

Fig. 8. Reduction of the indistinguishability of Game$_2$ and Game$_3$ to the CCA security of the underlying encryption scheme.

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

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

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

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

Since c is an encryption of $\overline{\mathsf{pk}}$, and $\mathsf{pk} \neq \overline{\mathsf{pk}}$ except with negligible probability, the perfect soundness of Π_{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 Σ as depicted in Figure 9.

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

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

```
$\underline{\mathsf{Sanit/Sign}'(m, \mathrm{MOD}, \mathrm{ADM}):}$
  $\rho \leftarrow \chi$
  $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}_{\mathsf{UFRK}}, \rho)$
  $\sigma_{\mathrm{FIX}} := \mathsf{SSign}_{\mathrm{FIX}}(\mathsf{sk}_{\mathrm{FIX}}, m_{\mathrm{FIX}})$
  $m' := \mathrm{MOD}(m)$
  $\sigma' := \mathcal{O}_2(m', \rho)$
  $c \leftarrow \mathsf{Enc}(\mathsf{ek}, \overline{\mathsf{pk}})$
  $C_{\mathsf{Sanit/Sign}} := C_{\mathsf{Sanit/Sign}} \cup \{(c, \rho)\}$
  $\mathsf{stmt} := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}')$
  $\tau \leftarrow \mathsf{S}_{\mathsf{PoK},1}(crs_{PoK}, \mathrm{T}_{\mathrm{PoK}}, \mathsf{stmt})$
  $\sigma := (\sigma_{\mathrm{FIX}}, \sigma', \mathrm{ADM}, \mathsf{pk}', c, \tau)$
  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 $(\sigma_{\mathrm{FIX}}, \sigma', \mathrm{ADM}, \mathsf{pk}', c, \tau)$.
  If $(c_i, \rho_i) \in C_{\mathsf{Sanit/Sign}}$ with $c_i = c$
    abort reduction, and output
      $(m, \sigma', \rho_i)$ as forgery
  $\widehat{\mathsf{pk}} \leftarrow \mathsf{Dec}(c)$
  $\mathsf{stmt} := (\mathsf{ek}, c, \widehat{\mathsf{pk}})$
  $\phi \leftarrow \mathsf{S}_{\mathsf{ZK},(}crs_{ZK}, \mathsf{stmt})$
  output $(\widehat{\mathsf{pk}}, \phi)$
```

Fig. 9. 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.

$\mathsf{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}'_{san}) = 1$ guarantees that $\mathsf{SVerify}(\mathsf{pk}', m, \sigma') = 1$ and that $\mathsf{V}_{\mathsf{PoK}}(crs_{PoK}, (\mathsf{ek}, c, \mathsf{pk}, \mathsf{pk}'_{san}, \mathsf{pk}'), \tau) = 1)$. 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 Sanit/Sign queries and and the Proof' oracle sets $\widehat{\mathsf{pk}} := \mathsf{pk}_{san}$ for ciphertexts $c \in C_{\mathsf{Sanit/Sign}}$.

$\underline{\mathsf{Game}_5 \approx \mathsf{Game}_6}$ 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 Sanit/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 Sanit/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 $\Sigma_{\mathrm{FIX}} = (\mathsf{SSetup}_{\mathrm{FIX}}, \mathsf{SGen}_{\mathrm{FIX}}, \mathsf{SSign}_{\mathrm{FIX}}, \mathsf{SVerify}_{\mathrm{FIX}})$ is a deterministic strongly existentially unforgeable signature scheme, 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

$$((m_i^0, \mathrm{MOD}_i^0, \sigma_i^0), (m_i^1, \mathrm{MOD}_i^1, \sigma_i^1))$$

be a query made by $\mathcal{A}$ to the LoRSanit oracle, where σ_i^b can be parsed as $(\sigma_{\text{FIX},i}^b, \sigma_i'^b, \text{ADM}_i^b, \mathsf{pk}'^b_i, c_i^b, \tau_i^b)$. The only difference between the two cases of the unlinkability experiment is the distribution of the answers to these queries if it holds that

$$\mathsf{Verify}(m_i^0, \sigma_i^0, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 1 \tag{25}$$
$$\mathsf{Verify}(m_i^1, \sigma_i^1, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 1 \tag{26}$$
$$\text{ADM}_i^0(\text{MOD}_i^0) \neq 0 \tag{27}$$
$$\text{ADM}_i^1(\text{MOD}_i^1) \neq 0 \tag{28}$$
$$\text{MOD}_i^0(m_i^0) = \text{MOD}_i^1(m_i^1) \tag{29}$$
$$\text{ADM}_i^0 = \text{ADM}_i^1 \tag{30}$$

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 rerandomizability 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

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

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 LoRSanit such that in addition to Equation 25 through Equation 30 it holds that

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

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

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

Now, consider reduction $\mathcal{B}_9$, depicted in Figure 10 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 $\mathsf{Unlinkability}_{\mathcal{A}}^{\text{SanS}}(\kappa)$ unless query occurs. Whenever query occurs, it holds because of Equation 33 that $m_{\text{FIX}}^0 = m_{\text{FIX}}^1$. Further, since Σ is deterministic, it holds that

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

Together with Equation 25 and Equation 26 it thus follows that

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

and therefore

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

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

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

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

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

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

Fig. 10. Description of reduction $\mathcal{B}_9$, reducing the occurence of event query in the unlinkability experiment of SanS against the s-EUF security of Σ.

5 Instantiating the Construction

We instantiate our generic construction with compatible and efficient instantiations. For the two signature schemes, we choose standard Schnorr signatures as defined in Definition 11 for Σ, as well as a derandomized version of Schnorr signatures for Σ_{FIX}[3]. The encryption scheme and proof systems are instantiated with the Cramer Shoup encryption scheme [13], and Σ-protocols that we convert into a non-interactive zero-knowledge proof via the Fiat-Shamir transform [14]. The Cramer Shoup encryption scheme is defined as follows:

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 the two non-interactive zero-knowledge proof systems. We instantiate these NIZK here with specific Fiat-Shamir transformed [14] Σ-protocols specified below.

Σ-Protocol for Encryption of Public Key In this section we show how to instantiate the Proof of Knowledge for the language $\mathcal{L}_1$. The statement that we want to prove in our concrete instantiation looks as follows:

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

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

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 [24]. We then formulate the complete proof using standard parallel composition techniques, first introduced in [11,12]. The complete protocol is depicted in Figure 11. 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 $(T_0, \ldots, T_5)$ 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.

Σ-Protocol for Proof of Decryption In this paragraph we show how to instantiate the proof of knowledge for the language $\mathcal{L}_2$. We prove the following statement:

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

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

Again, for the concrete instantiation in Figure 12 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.

[3] Note, that while the original security proof[22,23] for Schnorr signatures only proves standard existential unforgeability, it can be easily adapted to prove strong existential unforgeability

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

$$\begin{array}{ccc}
\underline{\text{Prover } (\mathsf{stmt}, i, \omega, \rho)} & & \underline{\text{Verifier } (\mathsf{stmt})} \\
(T_i, z_i, s_i) \leftarrow \mathsf{S}_{\Sigma}(1^{\kappa}) & & \\
u_{1-i}, u_2 \leftarrow \mathbb{Z}_q & & \\
T_{1-i} := g_1^{u_{1-i}} & & \\
T_2 := g_1^{u_2} \quad T_3 := g_2^{u_2} & & \\
T_4 := (cd^{\alpha})^{u_2} \quad T_5 := h^{u_2} & \xrightarrow{T_0, T_1, T_2, T_3, T_4, T_5} & \\
 & & z \leftarrow \mathbb{Z}_q \\
 & \xleftarrow[z]{} & \\
z_{1-i} := z - z_i & & \\
s_{1-i} := \rho \cdot z_{1-i} + u_{1-i} & & \\
s_2 := \omega \cdot z + u_2 & \xrightarrow{s_0, s_1, s_2, z_0} & \\
 & & z_1 := z - z_0 \\
 & & g_1^{s_0} \overset{?}{=} T_0 \cdot (\frac{\mathsf{pk}'}{\mathsf{pk}})^{z_0} \\
 & & g_1^{s_1} \overset{?}{=} T_1 \cdot (\frac{\mathsf{pk}'}{\mathsf{pk}_{san}})^{z_1} \\
 & & g_1^{s_2} \overset{?}{=} T_2 \cdot c_1^z \\
 & & g_2^{s_2} \overset{?}{=} T_3 \cdot c_2^z \\
 & & (cd^{\alpha})^{s_2} \overset{?}{=} T_4 \cdot c_4^z \\
 & & \frac{h^{s_2}}{g_1^{s_0+s_1}} \overset{?}{=} \frac{T_5}{T_0 T_1} \cdot (\frac{c_3}{\mathsf{pk}'})^z
\end{array}$$

Fig. 11. Σ-Protocol for Encryption of Public Key

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

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

Fig. 12. Σ-Protocol for Proof of Decryption

References

1. 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, Berlin, Germany. 1, 1.3
2. 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, Berlin, Germany. 1, 1.2
3. 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. 3, A.1
4. 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, Berlin, Germany. 1, 1.3, 2, 2.1, 2.1, 2.2, 4.2
5. 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, Berlin, Germany. 1, 1.2, 1, 1.2, 1.2, 2, 2, 1.3, 2, 2.1, 2.2, 4.2
6. Christina Brzuska, Henrich C. Pöhls, and Kai Samelin. Non-interactive public accountability for sanitizable signatures. In *Public Key Infrastructures, Services and Applications*, volume 7868 of *Lecture Notes in Computer Science*, pages 178–193. Springer Berlin Heidelberg, 2013. 1.3
7. Christina Brzuska, Henrich C. Pöhls, and Kai Samelin. Efficient and perfectly unlinkable sanitizable signatures without group signatures. In *Public Key Infrastructures, Services and Applications*, Lecture Notes in Computer Science, pages 12–30. Springer Berlin Heidelberg, 2014. 1.3
8. 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, Berlin, Germany. 3, A.2
9. 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, Berlin, Germany. 1.3
10. 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, Berlin, Germany. 1.3
11. 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, Berlin, Germany. 5
12. 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, Berlin, Germany. 5
13. 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, Berlin, Germany. 5
14. 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, Berlin, Germany. 1.1, 5, 5
15. 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, Berlin, Germany. 3.1
16. 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, Berlin, Germany. 1.2, 1, 1.2, 1.2, 2
17. 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, Berlin, Germany. 1.2, 1, 1.2, 1.2, 2, 2

18. 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, Berlin, Germany. 1.1, 3.1, 3.1, 12, 13, 3.1
19. Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. *Journal of Cryptology*, 25(3):484–527, July 2012. 1.1, 3.1, 3.1, 12, 13, 3.1
20. 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, Berlin, Germany. 1
21. Shigeo Mitsunari, Ryuichi Saka, and Masao Kasahara. A new traitor tracing. *IEICE Transactions*, E85-A(2):481–484, February 2002. 1
22. 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, Berlin, Germany. 3.1, 3
23. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000. 3.1, 3
24. 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, Berlin, Germany. 1.1, 3.1, 3.1, 5
25. Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991. 1.1, 3.1, 3.1
26. 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, Berlin, Germany. 1

A Insecure Signature Schemes Under Rerandomized Keys

A.1 Boneh Boyen

The scheme by Boneh and Boyen [3] works in a bilinear groups setting and is proven 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 rerandomized 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 proven existentially unforgeable and has perfectly rerandomizable keys it is trivially forgeable under rerandomized keys. The attack is very simple: 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 $\sigma = (r, s)$. It it finally 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 $\mathcal{A}$:

$$
\begin{aligned}
& & e(s, u \cdot g_2^{m'} \cdot v^r) &= e(g_1, g_2) & (35)\\
&\Leftrightarrow & e(s, g_2^{x+m+\rho_1+yr}) &= e(g_1, g_2) & (36)\\
&\Leftrightarrow & e(g_1^{\frac{1}{(x+\rho_1)+m+yr}}, g_2^{x+\rho_1+m+yr}) &= e(g_1, g_2) & (37)\\
&\Leftrightarrow & e(g_1, g_2)^{\frac{x+\rho_1+m+yr}{x+\rho_1+m+yr}} &= e(g_1, g_2) & (38)\\
&\Leftrightarrow & e(g_1, g_2) &= e(g_1, g_2) & (39)
\end{aligned}
$$

Further, the only message queried to the signing oracle is m, and $m' \neq m$. Therefore, it follows that $\mathcal{A}$ wins the UFRK game with probability 1.

A.2 Camenisch Lysyanskaya

The signature scheme by Camenisch and Lysyanskaya [8] works in a symmetric bilinear groups setting and is proven 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 rerandomized multiplicatively[4]. 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 $(X', Y') := (X^{\rho_1}, Y^{\rho_2})$.

Again, this scheme is proven existentially unforgeable and has perfectly rerandomizable keys. Nevertheless it also is trivially forgeable under rerandomized keys. The attack is very simple: 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 verification equation actually holds for the output of $\mathcal{A}$. For the first check equation we have:

$$\begin{aligned}
& & e(a, Y) &= e(g, b') & (40)\\
&\Leftrightarrow & e(a, g^y) &= e(g, b^{(\rho_2^{-1})}) & (41)\\
&\Leftrightarrow & e(g^y, a) &= e(g, a^{(y\rho_2)\cdot\rho_2^{-1}}) & (42)\\
&\Leftrightarrow & e(g^y, a) &= e(g, a^y) & (43)\\
&\Leftrightarrow & e(g, a)^y &= e(g, a)^y. & (44)
\end{aligned}$$

For the second check equation we have:

$$\begin{aligned}
& & e(X, a) \cdot e(X, b')^{m'} &= e(g, c) & (45)\\
&\Leftrightarrow & e(g^x, a) \cdot e(g^x, b^{\rho_2^{-1}})^{m\cdot\rho_2} &= e(g, a^{x+mxy\rho_2}) & (46)\\
&\Leftrightarrow & e(g, a)^x \cdot e(g^x, a^{y\rho_2\rho_2^{-1}})^{m\rho_2} &= e(g, a)^{x+mxy\rho_2} & (47)\\
&\Leftrightarrow & e(g, a)^x \cdot e(g, a)^{mxy\rho_2} &= e(g, a)^{x+mxy\rho_2} & (48)\\
&\Leftrightarrow & e(g, a)^{x+mxy\rho_2} &= e(g, a)^{x+mxy\rho_2}. & (49)
\end{aligned}$$

Further, 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 UFRK game with probability 1.

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