Unlinkable and Strongly Accountable Sanitizable Signatures from Verifiable Ring Signatures

Xavier Bultel, Pascal Lafourcade

Unlinkable and Strongly Accountable Sanitizable Signatures from Verifiable Ring Signatures*

Xavier Bultel[1,2] and Pascal Lafourcade[1,2]

[1] CNRS, UMR 6158, LIMOS, F-63173 Aubière, France
[2] Université Clermont Auvergne, BP 10448, 63000 Clermont-Ferrand, France

Abstract. An *Unlinkable Sanitizable Signature* scheme (USS) allows a sanitizer to modify some parts of a signed message in such away that nobody can link the modified signature to the original one. A *Verifiable Ring Signature* scheme (VRS) allows the users to sign messages anonymously within a group where a user can prove *a posteriori* to a verifier that it is the author of a given signature. In this paper, we first revisit the notion of VRS: we improve the proof capabilities of the users, we give a complete security model for VRS and we give an efficient and secure scheme called EVeR. Our main contribution is GUSS, a *Generic USS* based on a VRS scheme and an unforgeable signature scheme. We show that GUSS instantiated with EVeR and Schnorr's signature is twice as efficient as the best USS scheme of the literature. Moreover, we propose a stronger definition of *accountability*: an USS is *accountable* when the signer can prove whether a signature is sanitized. We formally define the notion of *strong accountability* where the sanitizer can also prove the origin of a signature. We show that the notion of strong accountability is important in practice. Finally, we prove the security properties of GUSS (including strong accountability) and EVeR under the Decisional Diffie-Hellman (DDH) assumption in the random oracle model.

1 Introduction

Sanitizable Signatures (SS) were introduced by Ateniese *et al.* [1], but similar primitives were independently proposed in [31]. In this primitive, a *signer* allows a proxy (called the *sanitizer*) to modify some parts of a signed message. For example, a magistrate wishes to delegate the power to summon someone to the court to his secretary. He signs the message "*Franz* is summoned to court for an interrogation on *Monday*" and gives the signature to his secretary, where "Franz" and "Monday" are sanitizable and the other parts are fixed. Thus, in order to summon Joseph K. on Saturday in the name of the magistrate, the secretary can change the signed message into "*Joseph K.* is summoned to the court for an interrogation on *Saturday*".

Ateniese *et al.* in [1] proposed some applications of this primitive in privacy of health data, authenticated media streams and reliable routing information. They also introduced five security properties formalized by Brzuska *et al.* in [8]:
Unforgeability: no unauthorized user can generate a valid signature.

* This research was conducted with the support of the "Digital Trust" Chair from the University of Auvergne Foundation.

Immutability: the sanitizer cannot transform a signature from an unauthorized message.
Privacy: no information about the original message is leaked by a sanitized signature.
Transparency: nobody can say if a signature is sanitized or not.
Accountability: the signer can prove whether a signature is sanitized.

Finally, in [9] the authors point out a non-studied but relevant property called *unlinkability*: a scheme is unlinkable when it is not possible to link a sanitized signature to the original one. The authors give a generic unlinkable scheme based on group signatures. In 2016, Fleischhacker *et al.* [21] give a more efficient construction based on signatures with re-randomizable keys.

On the other hand, ring signature is a well-studied cryptographic primitive introduced by Rivest *et al.* in [29], where some users can sign anonymously within a group of users. Such a scheme is *verifiable* [28] when any user can prove that he is the signer of a given message. In this paper, we improve the properties of VRS by increasing the proof capabilities of the users. We also give an efficient VRS scheme called EVeR and a generic unlinkable sanitizable signature scheme called GUSS that uses a verifiable ring signature. We also show that the definition of accountability is too weak for practical use and we propose a stronger definition.

Contributions: Existing VRS schemes allow any user to prove that he is the signer of a given message. We extend the definition of VRS to allow a user to prove that he is not the signer of a given message. We give a formal security model for VRS that takes into account this property. We first extend the classical security properties of ring signatures to verifiable ring signatures, namely the *unforgeability* (no unauthorized user can forge a valid signature) and the *anonymity* (nobody can distinguish who is the signer in the group). In addition we define the *accountability* (if a user signs a message then he cannot prove that he is not the signer) and the *non-seizability* (a user cannot prove that he is the signer of a message if it is not true, and a user cannot forge a message such that the other users cannot prove that they are not the signers). To the best of our knowledge, it is the first time that formal security models are proposed for VRS. We also design an efficient secure VRS scheme under the DDH assumption in the random oracle model.

The definition of accountability for SS given in [8, 9, 21] considers that the signer can prove the origin of a signature (signer or sanitizer) by using a proof algorithm such that:

1. The signer cannot forge a signature and a proof that the signature has been forged by the sanitizer.
2. The sanitizer cannot forge a signature such that the proof algorithm blames the signer.

The proof algorithm requires the secret key of the signer. To show that this definition is too weak, we consider a signer who cannot prove the origin of a litigious signature. The signer claims that he lost his secret key because of problems with his hard drive. There is no way to verify whether the signer is lying. Unfortunately, without his secret key, the signer cannot generate the proof for the litigious signature, hence nobody can judge whether the signature is sanitized or not. Depending on whether the signer is lying, there is a risk of accusing the signer or the sanitizer wrongly. To solve this problem, we

add a second proof algorithm that allows the sanitizer to prove the origin of a signature. To achieve *strong accountability*, the two following additional properties are required:

1. The sanitizer cannot sanitize a signature σ and prove that σ is not sanitized.
2. The signer cannot forge a signature such that the sanitizer proof algorithm accuses the sanitizer.

The main contribution of this paper is to propose an efficient and generic unlinkable SS scheme called GUSS. This scheme is instantiated by a VRS and an unforgeable signature scheme. It is the first SS scheme that achieves strong accountability. We compare GUSS with the other schemes of the literature:

Brzuska *et al.* [9]. This scheme is based on group signatures. Our scheme is built on the same model, but it uses ring signatures instead of group signatures. The main advantage of group signatures is that the size of the signature is not proportional to the size of the group. However, for small groups, ring signatures are much more efficient than group signatures. Since the scheme of Brzuska *et al.* and GUSS uses group/ring signatures for groups of two users, GUSS is much more practical for an equivalent level of genericity.

Fleischhacker *et al.* [21]. This scheme is based on *signatures with re-randomizable keys*. It is generic, however it uses different tools that must have special properties to be compatible with each other. To the best of our knowledge, it is the most efficient scheme of the literature. GUSS instantiated with EVeR and Schnorr's signature is twice as efficient as the best instantiation of this scheme. In Fig. A, we compare the efficiency of each algorithm of our scheme and the scheme of Fleischhacker *et al.*.

Lai *et al.* [26]. Recently, Lai *et al.* proposed a USS that is secure in the standard model. This scheme uses pairings and is much less efficient than the scheme of Fleischhacker *et al.*, so this scheme is much less efficient than our scheme. In their paper [26], Lai *et al.* give a comparison of the efficiency of the three schemes of the literature.

	SiGen	SaGen	Sig	San	Ver	SiProof	SiJudge	**Total**	pk	spk	sk	ssk	σ	π	**Total**
[21]	7	1	15	14	17	23	6	**83**	7	1	14	1	14	4	**41**
GUSS	2	1	8	7	10	3	4	**35**	2	1	2	1	12	5	**23**

Fig. 1. Comparison of GUSS and the scheme of Fleischhacker *et al.*: The first six columns give the number of exponentiations of each algorithms of both schemes, namely the key generation algorithm of the signer (SiGen) and the sanitizer (SaGen), the signature algorithm (Sig), the verification algorithm (Ver), the sanitize algorithm (San), the proof algorithm (SiProof) and the judge algorithm (SiJudge). The last six columns give respectively the size of the public key of the signer (pk) and the sanitizer (spk), the size of the secret key of the signer (sk) and the sanitizer (ssk), the size of a signature (σ) and the size of a proof (π) outputted by SiProof. This size is measured in elements of a group $\mathbb{G}$ of prime order. As in [21], for the sake of clarity, we do not distinguish between elements of $\mathbb{G}$ and elements of $\mathbb{Z}_p^*$. We consider the best instantiation of the scheme of Fleischhacker *et al.* given in [21]. In Appendix A, we give a detailed complexity evaluation of our schemes.

Related works: *Sanitizable Signatures* (SS) was first introduced by Ateniese *et al.* [1]. Later, Brzuska *et al.* gave formal security definitions [8] for *unforgeability*, *immutability*, *privacy*, *transparency* and *accountability*. *Unlinkability* was introduced and formally defined by Brzuska *et al.* in [9]. In [10], Brzuska *et al.* introduce an alternative definition of accountability called *non-interactive public accountability* where the capability to prove the origin of a signature is given to a third party. One year later, the same authors propose a stronger definition of unlinkability [11] and design a scheme that is both strongly unlinkable and non-interactively public accountable. However, non-interactive public accountability is not compatible with transparency. In this paper, we focus on schemes that are unlinkable, transparent and interactively accountable. To the best of our knowledge, there are only 3 schemes with these 3 properties, *i.e.* [9, 21, 26].

Some works focus on other properties of SS that we do not consider here, such as SS with multiple sanitizers [14], or SS where the power of the sanitizer is limited [13]. Finally, there exist other primitives that solve related but different problems such as homomorphic signatures [25], redactable signatures [7] or proxy signatures [22]. The differences between these primitives and sanitizable signatures are detailed in [21].

On the other hand, *Ring Signatures* (RS) [29] were introduced by Rivest *et al.* in 2003. Security models of this primitive were defined in [4]. *Verifiable Ring Signatures* (VRS) [28] were introduced in 2003 by Lv. RS allow the users to sign anonymously within a group, and VRS allow a user to prove that it is the signer of a given message. The authors of [28] give a VRS construction that is based on the discrete logarithm problem. Two other VRS schemes were proposed by Wand *et al.* [32] and by Changlung *et al.* [16]. The first one is based on the Nyberg-Rueppel signature scheme and the second one is a generic construction based on multivariate public key cryptosystems. In these three schemes, a user can prove that he is the signer of a given signature, however, he has no way to prove that he is not the signer, and it seems to be non-trivial to add this property to these schemes. Convertible ring signatures [27] are very close to verifiable ring signatures: they allow the signer of an anonymous (ring) signature to transform it into a standard signature (*i.e.* a *deanonymized* signature). It can be used as a verifiable ring signature because the deanonymized signature can be viewed as a proof that the user is the signer of a given message. However, in this paper we propose a stronger definition of VRS where a user also can prove that he is not the signer of a message, and this property cannot be achieved using convertible signatures.

A *Revocable-iff-Linked Ring Signature* (RLRS) [2] (also called *List Signature* [15]) is a kind of RS that has the following property: if a user signs two messages for the same *event-id*, then it is possible to link these signatures and the user's identity is publicly revealed. It can be used to design a VRS in our model: to prove whether he is the signer of a given message, the user signs a second message using the same event-id. If the two signatures are linked, then the judge is convinced that the user is the signer, else he is convinced that the user is not the signer. However, RLRS requires security properties that are too strong for VRS (linkability and traceability) and it would result in less efficient schemes.

Outline: In Section 2, we recall the standard cryptographic tools used in this paper. In Section 3 and Section 4, we present the formal definition and the security models

for verifiable ring signatures and unlinkable sanitizable signatures. In Section 5, we present our scheme EvER. Finally, in Section 6, we present our scheme GUSS, before concluding in Section 7.

2 Cryptographic Tools

We present the cryptographic tools used throughout this paper. We first recall the DDH assumption.

Definition 1 (DDH [5]). *Let $\mathbb{G}$ be a multiplicative group of prime order p and $g \in \mathbb{G}$ be a generator. Given an instance $h = (g^a, g^b, g^z)$ for unknown $a, b, z \xleftarrow{\$} \mathbb{Z}_p^*$, the* Decisional Diffie-Hellman *(DDH)* problem *is to decide whether $z = a \cdot b$ or not. The DDH* assumption *states that there exists no polynomial time algorithm that solves the DDH problem with a non-negligible advantage.*

In the following, we recall some notions about digital signatures.

Definition 2 ((Deterministic) Digital Signature (DS)). *A* Digital Signature *scheme $S = (\mathsf{D.Init}, \mathsf{D.Gen}, \mathsf{D.Sig}, \mathsf{D.Ver})$ is a tuple of four algorithms defined as follows:*

$\mathsf{D.Init}(1^k)$: *It returns a setup value* set.
$\mathsf{D.Gen}(\mathsf{set})$: *It returns a pair of signer public/private keys* $(\mathsf{pk}, \mathsf{sk})$.
$\mathsf{D.Sig}(m, \mathsf{sk})$: *It returns a signature σ of m using the key* sk.
$\mathsf{D.Ver}(\mathsf{pk}, m, \sigma)$: *It returns a bit b.*

S is said to be deterministic when the algorithm $\mathsf{D.Sig}$ *is deterministic. S is said to be correct when for any security parameter $k \in \mathbb{N}$, any message $m \in \{0,1\}^*$, any* $\mathsf{set} \leftarrow \mathsf{D.Init}(1^k)$ *and any* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{D.Gen}(\mathsf{set})$, $\mathsf{D.Ver}(\mathsf{pk}, m, \mathsf{D.Sig}(m, \mathsf{sk})) = 1$. *Moreover, such a scheme is* unforgeable *when no polynomial adversary wins the following experiment with non-negligible probability where $\mathsf{D.Sig}(\cdot, \mathsf{sk})$ is a signature oracle, q_S is the number of queries to this oracle and σ_i is the i^{th} signature computed by the signature oracle:*

$\mathbf{Exp}^{\mathsf{unf}}_{S,\mathcal{A}}(k)$:
$\mathsf{set} \leftarrow \mathsf{D.Init}(1^k)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{D.Gen}(\mathsf{set})$
$(m_*, \sigma_*) \leftarrow \mathcal{A}^{\mathsf{D.Sig}(\cdot, \mathsf{sk})}(\mathsf{pk})$
if $(\mathsf{D.Ver}(\mathsf{pk}, m_*, \sigma_*) = 1)$ *and* $(\forall\, i \in [\![1, q_S]\!],\ \sigma_i \neq \sigma_*)$
then return 1, *else return* 0

As it is mentioned in [21], any DS scheme can be changed into a deterministic scheme without loss of security using a pseudo random function, that can be simulated by a hash function in the random oracle model. The following scheme is the deterministic version of the well-known Schnorr's Signature scheme [30].

Scheme 1 (Deterministic Schnorr's Signature [30]) *This signature is defined by the following algorithms:*

D.Init(1^k): *It returns a setup value* $\mathsf{set} = (\mathbb{G}, p, g, H)$ *where* $\mathbb{G}$ *is a group of prime order* p*,* $g \in \mathbb{G}$ *and* $H : \{0,1\}^* \to \mathbb{Z}_p^*$ *is a hash function.*

D.Gen(set): *It picks* $\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_p^*$*, computes* $\mathsf{pk} = g^{\mathsf{sk}}$ *and returns* $(\mathsf{pk}, \mathsf{sk})$.

D.Sig(m, sk): *It computes the* $r = H(m||\mathsf{sk})$*,* $R = g^r$*,* $z = r + \mathsf{sk} \cdot H(R||m)$ *and returns* $\sigma = (R, z)$.

D.Ver(pk, m, σ): *It parses* $\sigma = (R, z)$*, if* $g^z = R \cdot \mathsf{pk}^{H(R||m)}$ *then it returns* 1*, else* 0.

This DS scheme is deterministic and unforgeable under the DL assumption in the random oracle model.

A Zero-Knowledge Proof (ZKP) [23] allows a prover knowing a witness w to convince a verifier that a statement s is in a given language without leaking any information. Such a proof is a Proof of Knowledge (PoK) [3] when the verifier is also convinced that the prover knows the witness w. We recall the definition of a non-interactive zero-knowledge proof of knowledge.

Definition 3 (NIZKP). *Let* $\mathcal{R}$ *be a binary relation and let* $\mathcal{L}$ *be a language such that* $s \in \mathcal{L} \Leftrightarrow (\exists w, (s, w) \in \mathcal{R})$*. A* non-interactive ZKP *(NIZKP) for the language* $\mathcal{L}$ *is a couple of algorithms* (Prove, Verify) *such that:*

Prove(s, w)**.** *This algorithm outputs a proof* π*.*
Verify(s, π)**.** *This algorithm outputs a bit* b*.*

A NIZKP proof has the following properties:

Completeness. *For any statement* $s \in \mathcal{L}$ *and the corresponding witness* w*, we have that* Verify$(s,$ Prove$(s, w)) = 1$.

Soundness. *There is no polynomial time adversary* $\mathcal{A}$ *such that* $\mathcal{A}(\mathcal{L})$ *outputs* (s, π) *such that* Verify$(s, \pi) = 1$ *and* $s \notin \mathcal{L}$ *with non-negligible probability.*

Zero-knowledge. *A proof* π leaks no information*, i.e. there exists a polynomial time algorithm* Sim *(called the* simulator*) such that* Prove(s, w) *and* Sim(s) *follow the same probability distribution.*

Moreover, such a proof is a proof of knowledge *when for any* $s \in \mathcal{L}$ *and any algorithm* $\mathcal{A}$*, there exists a polynomial time knowledge extractor* $\mathcal{E}$ *such that the probability that* $\mathcal{E}^{\mathcal{A}(s)}(s)$ *outputs a witness* w *such that* $(s, w) \in \mathcal{R}$ *given access to* $\mathcal{A}(s)$ *as an oracle is as high as the probability that* $\mathcal{A}(s)$ *outputs a proof* π *such that* Verify$(s, \pi) = 1$.

3 Formal Model of Verifiable Ring Signatures

We formally define the Verifiable Ring Signatures (VRS) and the corresponding security notions. A VRS is a ring signature scheme where a user can prove to a judge whether he is the signer of a message or not. It is composed of six algorithms. V.Init, V.Gen, V.Sig and V.Ver are defined as in the usual ring signature definitions. V.Gen generates public and private keys. V.Sig anonymously signs a message according to a set of public keys. V.Ver verifies the soundness of a signature. A VRS has two additional algorithms: V.Proof allows a user to prove whether he is the signer of a message or not, and V.Judge allows anybody to verify the proofs outputted by V.Proof.

Definition 4 (Verifiable Ring Signature (VRS)). *A* Verifiable Ring Signature *scheme is a tuple of six algorithms defined by:*

$\mathsf{V.Init}(1^k)$: *It returns a setup value* set.

$\mathsf{V.Gen}(\mathsf{set})$: *It returns a pair of signer public/private keys* $(\mathit{pk}, \mathit{sk})$.

$\mathsf{V.Sig}(L, m, \mathsf{sk})$: *This algorithm computes a signature* σ *using the key* sk *for the message* m *according to the set of public keys* L.

$\mathsf{V.Ver}(L, m, \sigma)$: *It returns a bit* b*: if the signature* σ *of* m *is valid according to the set of public key* L *then* $b = 1$*, else* $b = 0$.

$\mathsf{V.Proof}(L, m, \sigma, \mathsf{pk}, \mathsf{sk})$: *It returns a proof* π *for the signature* σ *of* m *according to the set of public key* L.

$\mathsf{V.Judge}(L, m, \sigma, \mathsf{pk}, \pi)$: *It returns a bit* b *or the bottom symbol* $\perp$*: if* $b = 1$ *(resp.* 0*) then* π *proves that* σ *was (resp. was not) generated by the signer corresponding to the public key* pk*. It outputs* $\perp$ *when the proof is not well formed.*

Unforgeability: We first adapt the unforgeability property of ring signatures [4] to VRS. Informally, a VRS is unforgeable when no adversary is able to forge a signature for a ring of public keys without any corresponding secret key. In this model, the adversary has access to a signature oracle $\mathsf{V.Sig}(\cdot,\cdot,\cdot)$ (that outputs signatures of chosen messages for chosen users in the ring) and a proof oracle $\mathsf{V.Proof}(\cdot,\cdot,\cdot,\cdot,\cdot)$ (that outputs proofs as the algorithm $\mathsf{V.Proof}$ for chosen signatures and chosen users). The adversary succeeds when it outputs a valid signature that was not already generated by the signature oracle.

Definition 5 (Unforgeability). *Let* P *be a VRS and* n *be an integer. Let the two following oracles be:*

$\mathsf{V.Sig}(\cdot,\cdot,\cdot)$**:** *On input* (L, l, m)*, if* $1 \leq l \leq n$ *then it runs* $\sigma \leftarrow \mathit{V.Sig}(L, \mathit{sk}_l, m)$ *and returns* σ*, else it returns* $\perp$.

$\mathsf{V.Proof}(\cdot,\cdot,\cdot,\cdot,\cdot)$**:** *On input* (L, m, σ, l)*, if* $1 \leq l \leq n$ *then this proof oracle runs* $\pi \leftarrow \mathit{V.Proof}(L, m, \sigma, \mathit{pk}_l, \mathit{sk}_l)$ *and returns* π*, else it returns* $\perp$.

P *is* n*-*unf *secure when for any polynomial time adversary* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *wins the following experiment is negligible, where* q_S *is the number of calls to the oracle* $\mathit{V.Sig}(\cdot,\cdot,\cdot)$ *and* σ_i *is the* i^{th} *signature outputted by the oracle* $\mathit{V.Sig}(\cdot,\cdot,\cdot)$*:*

$\mathbf{Exp}^{n\text{-}\mathbf{unf}}_{P,\mathcal{A}}(k)$*:*
$\mathit{set} \leftarrow \mathit{V.Init}(1^k)$
$\forall 1 \leq i \leq n, (\mathit{pk}_i, \mathit{sk}_i) \leftarrow \mathit{V.Gen}(\mathit{set})$
$(L_*, \sigma_*, m_*) \leftarrow \mathcal{A}^{\mathit{V.Sig}(\cdot,\cdot,\cdot),\mathit{V.Proof}(\cdot,\cdot,\cdot,\cdot,\cdot)}(\{\mathit{pk}_i\}_{1\leq i\leq n})$
if $\mathit{V.Ver}(L_*, \sigma_*, m_*) = 1$ *and* $L_* \subseteq \{\mathit{pk}_i\}_{1\leq i\leq n}$ *and* $\forall\, i \in [\![1, q_S]\!]$, $\sigma_i \neq \sigma_*$
then return 1*, else return* 0

P *is* unforgeable *when it is* $t(k)$*-*unf *secure for any polynomial* t.

Annonymity: We adapt the anonymity property of ring signatures [4] to VRS. Informally, a VRS is anonymous when no adversary is able to link a signature to the corresponding user. The adversary has access to the signature oracle and the proof oracle. During a first phase, it chooses two honest users in the ring, and in the second phase, it has access to a challenge oracle $\mathsf{LRSO}_\mathsf{b}(d_0, d_1, \cdot, \cdot)$ that outputs signatures of chosen messages using the secret key of one of the two chosen users. The adversary succeeds

if he guesses which user is chosen by the challenge oracle. Note that if the adversary uses the proof oracle on the signatures generated by the challenge oracle then he loses the experiment.

Definition 6 (Anonymity). *Let P be a VRS and let n be an integer. Let the following oracle be:*

$\mathsf{LRSO}_\mathsf{b}(d_0, d_1, \cdot, \cdot)$**:** *On input (m, L), if $\{\mathsf{pk}_{d_0}, \mathsf{pk}_{d_1}\} \subseteq L$ then this oracle runs $\sigma \leftarrow \mathsf{V.Sig}(L, \mathsf{sk}_{d_b}, m)$ and returns σ, else it returns $\perp$.*

P is n-ano secure when for any polynomial time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the probability that $\mathcal{A}$ wins the following experiment is negligibly close to $1/2$, where $\mathsf{V.Sig}(\cdot,\cdot,\cdot)$ and $\mathsf{V.Proof}(\cdot,\cdot,\cdot,\cdot,\cdot)$ are defined as in Def. 5 and where q_S (resp. q_P) is the number of calls to the oracle $\mathsf{V.Sig}(\cdot,\cdot,\cdot)$ (resp. $\mathsf{V.Proof}(\cdot,\cdot,\cdot,\cdot,\cdot)$), $(L_i, m_i, \sigma_i, l_i)$ is the i^{th} query sent to oracle $\mathsf{V.Proof}(\cdot,\cdot,\cdot,\cdot,\cdot)$ and σ'_j is the j^{th} signature outputted by the oracle $\mathsf{LRSO}_\mathsf{b}(d_0, d_1, \cdot, \cdot)$:

$\mathbf{Exp}^{n\text{-}\mathbf{ano}}_{P,\mathcal{A}}(k)$:
$\mathsf{set} \leftarrow \mathsf{V.Init}(1^k)$
$\forall 1 \leq i \leq n, (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{V.Gen}(\mathsf{set})$
$(d_0, d_1) \leftarrow \mathcal{A}_1^{\mathsf{V.Sig}(\cdot,\cdot,\cdot),\mathsf{V.Proof}(\cdot,\cdot,\cdot,\cdot,\cdot)}(\{\mathsf{pk}_i\}_{1\leq i\leq n})$
$b \xleftarrow{\$} \{0,1\}$
$b_* \leftarrow \mathcal{A}_2^{\mathsf{V.Sig}(\cdot,\cdot,\cdot),\mathsf{V.Proof}(\cdot,\cdot,\cdot,\cdot,\cdot),\mathsf{LRSO}_\mathsf{b}(d_0,d_1,\cdot,\cdot)}(\{\mathsf{pk}_i\}_{1\leq i\leq n})$
if $(b = b_*)$ *and* $(\forall\, i, j \in [\![1, \max(q_S, q_P)]\!]$, $(\sigma_i \neq \sigma'_j)$ *or* $(l_i \neq d_0$ *and* $l_i \neq d_1))$
then return 1*, else return* 0

P *is* anonymous *when it is $t(k)$-ano secure for any polynomial t.*

Accountability: We consider an adversary that has access to a proof oracle and a signature oracle. A VRS is accountable when no adversary is able to forge a signature σ (that was not outputted by the signature oracle) together with a proof that it is not the signer of σ. Note that the ring of σ must contain at most one public key that does not come from an honest user, thus the adversary knows at most one secret key that corresponds to a public key in the ring.

Definition 7 (Accountability). *Let P be a VRS and let n be an integer. P is n-acc secure when for any polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following experiment is negligible, where $\mathsf{V.Sig}(\cdot,\cdot,\cdot)$ and $\mathsf{V.Proof}(\cdot,\cdot,\cdot,\cdot,\cdot)$ are defined as in Def. 5 and where q_S is the number of calls to the oracle $\mathsf{V.Sig}(\cdot,\cdot,\cdot)$ and σ_i is the i^{th} signature outputted by the oracle $\mathsf{V.Sig}(\cdot,\cdot,\cdot)$:*

$\mathbf{Exp}^{n\text{-}\mathbf{acc}}_{P,\mathcal{A}}(k)$:
$\mathsf{set} \leftarrow \mathsf{V.Init}(1^k)$
$\forall 1 \leq i \leq n, (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{V.Gen}(\mathsf{set})$
$(L_*, m_*, \sigma_*, \mathsf{pk}_*, \pi_*) \leftarrow \mathcal{A}^{\mathsf{V.Sig}(\cdot,\cdot,\cdot),\mathsf{V.Proof}(\cdot,\cdot,\cdot,\cdot,\cdot)}(\{\mathsf{pk}_i\}_{1\leq i\leq n})$
if $(L \subseteq \{\mathsf{pk}_i\}_{1\leq i\leq n} \cup \{\mathsf{pk}_*\})$ *and* $(\mathsf{V.Ver}(L_*, \sigma_*, m_*) = 1)$ *and*
$(\mathsf{V.Judge}(L_*, m_*, \sigma_*, \mathsf{pk}_*, \pi_*) = 0)$ *and* $(\forall\, i \in [\![1, q_S]\!]$, $\sigma_i \neq \sigma_*)$
then return 1*, else return* 0

P *is* accountable *when it is $t(k)$-acc secure for any polynomial t.*

Non-seizability: We distinguish two experiments for this property: the first experiment, denoted non-sei-1, considers an adversary that has access to a proof oracle and a signature oracle. Its goal is to forge a valid signature with a proof that the signer is another user in the ring.

Definition 8 (n**-non-sei-1 experiment).** *Let P be a SS. P is n-non-sei-1 secure when for any polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following experiment is negligible, where* $\mathsf{V.Sig}(\cdot,\cdot,\cdot)$ *and* $\mathsf{V.Proof}(\cdot,\cdot,\cdot,\cdot,\cdot)$ *and where q_S is the number of calls to the oracle* $\mathsf{V.Sig}(\cdot,\cdot,\cdot)$ *and (L_i, l_i, m_i) (resp. σ_i) is the i^{th} query to the oracle* $\mathsf{V.Sig}(\cdot,\cdot,\cdot)$ *(resp. signature outputted by this oracle):*

$\mathbf{Exp}_{P,\mathcal{A}}^{n\text{-non-sei-1}}(k)$*:*
$\mathsf{set} \leftarrow \mathsf{V.Init}(1^k)$
$\forall 1 \leq i \leq n, (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{V.Gen}(\mathsf{set})$
$(L_*, m_*, \sigma_*, l_*, \pi_*) \leftarrow \mathcal{A}^{\mathsf{V.Sig}(\cdot,\cdot,\cdot),\mathsf{V.Proof}(\cdot,\cdot,\cdot,\cdot,\cdot)}(\{\mathsf{pk}_i\}_{1\leq i\leq n})$
$\pi \leftarrow \mathsf{V.Proof}(L_*, m_*, \sigma_*, \mathsf{pk}, \mathsf{sk})$
if $(\mathsf{V.Ver}(L_*, \sigma_*, m_*) = 1)$ *and*
$\quad(\mathsf{V.Judge}(L_*, m_*, \sigma_*, \mathsf{pk}_{l_*}, \pi_*) = 1)$ *and*
$\quad(\forall\, i \in [\![1, q_S]\!],\ (L_i, l_i, m_i, \sigma_i) = (L_*, l_*, m_*, \sigma_*))$
then return 1*, else return* 0

The second experiment, denoted non-sei-2, considers an adversary that has access to a proof oracle and a signature oracle and that receives the public key of a honest user as input. The goal of the adversary is to forge a signature σ such that the proof algorithm ran by the honest user returns a proof that σ was computed by the honest user (*i.e.* the judge algorithm returns 1) or a non-valid proof (*i.e.* the judge algorithm returns $\perp$). Moreover, the signature σ must not come from the signature orale.

Definition 9 (Non-seizability). *Let P be a VRS and n be an integer. P is n-non-sei-2 secure when for any polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following experiment is negligible, where* $\mathsf{V.Sig}(\cdot,\cdot,\cdot)$ *and* $\mathsf{V.Proof}(\cdot,\cdot,\cdot,\cdot,\cdot)$ *are defined as in Def. 5 and where q_S is the number of calls to the oracle* $\mathsf{V.Sig}(\cdot,\cdot,\cdot)$ *and σ_i is the i^{th} signature outputted by the oracle* $\mathsf{V.Sig}(\cdot,\cdot,\cdot)$*:*

$\mathbf{Exp}_{P,\mathcal{A}}^{n\text{-non-sei-2}}(k)$*:*
$\mathsf{set} \leftarrow \mathsf{V.Init}(1^k)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{V.Gen}(\mathsf{set})$
$(L_*, m_*, \sigma_*) \leftarrow \mathcal{A}^{\mathsf{V.Sig}(\cdot,\cdot,\cdot),\mathsf{V.Proof}(\cdot,\cdot,\cdot,\cdot,\cdot)}(\mathsf{pk})$
$\pi \leftarrow \mathsf{V.Proof}(L_*, m_*, \sigma_*, \mathsf{pk}, \mathsf{sk})$
if $(\mathsf{V.Ver}(L_*, \sigma_*, m_*) = 1)$ *and*
$\quad(\mathsf{V.Judge}(L_*, m_*, \sigma_*, \mathsf{pk}_*, \pi_*) \neq 0)$ *and* $(\forall\, i \in [\![1, q_S]\!],\ \sigma_i \neq \sigma_*)$
then return 1*, else return* 0

P is non-seizable *when it is both $t(k)$-non-sei-1 and $t(k)$-non-sei-2 secure for any polynomial t.*

4 Formal Model of Sanitizable Signature

We give the formal definition and security properties of the sanitizable signature primitive. Compared to the previous definitions where only the signer can prove the origin of

a signature, our definition introduces algorithms that allow the sanitizer to prove the origin of a signature. Moreover, in addition to the usual security models of [8], we present two new security experiments that improve the accountability definition.

A SS scheme contains 10 algorithms. Init outputs the setup values. SiGen and SaGen generate respectively the signer and the sanitizer public/private keys. As in classical signature schemes, the algorithms Sig and Ver allow the users to sign a message and to verify a signature. However, the signatures are computed using a sanitizer public key and an admissible function ADM. The algorithm San allows the sanitizer to transform a signature of a message m according to a modification function MOD: if MOD is admissible according to the admissible function (*i.e.* $\text{MOD}(\text{ADM}) = 1$) this algorithm returns a signature of the message $m' = \text{MOD}(m)$.

SiProof allows the signer to prove whether a signature is sanitized or not. Proofs outpoutted by this algorithm can be verified by anybody using the algorithm SiJudge. Finally, algorithms SaProof and SaJudge have the same functionalities as SiProof and SiJudge, but the proofs are computed from the secret parameters of the sanitizer instead of the signer.

Definition 10 (Sanitizable Signature (SS)). *A* Sanitizable Signature *scheme is a tuple of* 10 *algorithms defined as follows:*

$\mathsf{Init}(1^k)$: *It returns a setup value* ***set****.*

$\mathsf{SiGen}(\mathsf{set})$: *It returns a pair of signer public/private keys* (***pk***, ***sk***)*.*

$\mathsf{SaGen}(\mathsf{set})$: *It returns a pair of sanitizer public/private keys* (***spk***, ***ssk***)*.*

$\mathsf{Sig}(m, \mathsf{sk}, \mathsf{spk}, \text{ADM})$: *This algorithm computes a signature* σ *from the message* m *using the secret key* ***sk****, the sanitizer public key* ***spk*** *and the admissible function* ADM*. Note that we assume that* ADM *can be efficiently recovered from any signature as in the definition of Fleischhacker et al. [21].*

$\mathsf{San}(m, \text{MOD}, \sigma, \mathsf{pk}, \mathsf{ssk})$: *Let the admissible function* ADM *according to the signature* σ*. If* $\text{ADM}(\text{MOD}) = 1$ *then this algorithm returns a signature* σ' *of the message* $m' = \text{MOD}(m)$ *using the signature* σ*, the signer public key* ***pk*** *and the sanitizer secret key* ***ssk****. Else it returns* $\perp$*.*

$\mathsf{Ver}(m, \sigma, \mathsf{pk}, \mathsf{spk})$: *It returns a bit* b*: if the signature* σ *of* m *is valid for the two public keys* ***pk*** *and* ***spk*** *then* $b = 1$*, else* $b = 0$*.*

$\mathsf{SiProof}(\mathsf{sk}, m, \sigma, \mathsf{spk})$: *It returns a signer proof* π_{si} *for the signature* σ *of* m *using the signer secret key* ***sk*** *and the sanitizer public key* ***spk****.*

$\mathsf{SaProof}(\mathsf{ssk}, m, \sigma, \mathsf{pk})$: *It returns a sanitizer proof* π_{sa} *for the signature* σ *of* m *using the sanitizer secret key* ***ssk*** *and the signer public key* ***pk****.*

$\mathsf{SiJudge}(m, \sigma, \mathsf{pk}, \mathsf{spk}, \pi_{\mathsf{si}})$: *It returns a bit* d *or the bottom symbol* $\perp$*: if* π_{si} *proves that* σ *comes from the signer corresponding to the public key* ***pk*** *then* $d = 1$*, else if* π_{si} *proves that* σ *comes from the sanitizer corresponding to the public key* ***spk*** *then* $d = 0$*, else the algorithm outputs* $\perp$*.*

$\mathsf{SaJudge}(m, \sigma, \mathsf{pk}, \mathsf{spk}, \pi_{\mathsf{sa}})$: *It returns a bit* d *or the bottom symbol* $\perp$*: if* π_{sa} *proves that* σ *comes from the signer corresponding to the public key* ***pk*** *then* $d = 1$*, else if* π_{sa} *proves that* σ *comes from the sanitizer corresponding to the public key* ***spk*** *then* $d = 0$*, else the algorithm outputs* $\perp$*.*

As it is mentioned in Introduction, SS schemes have the following security properties: *unforgeability*, *immutability*, *privacy*, *transparency* and *accountability*. In [8]

authors show that if a scheme has the *immutability*, the *transparency* and the *accountability* properties, then it has the *unforgeability* and the *privacy* properties. Hence we do not need to prove these two properties, so we do not recall their formal definitions.

Immutability: A SS is immutable when no adversary is able to sanitize a signature without the corresponding sanitizer secret key or to sanitize a signature using a modification function that is not admissible (*i.e.* $\text{ADM}(\text{MOD}) = 0$). To help him, the adversary has access to a signature oracle $\mathsf{Sig}(.,\mathsf{sk},.,.)$ and a proof oracle $\mathsf{SiProof}(\mathsf{sk},.,.,.)$.

Definition 11 (Immutability [8]). *Let the following oracles be:*
$\mathsf{Sig}(.,\mathsf{sk},.,.)$**:** *On input* $(m, \text{ADM}, \mathit{spk})$*, this oracle returns* $\mathit{Sig}(m, \mathit{sk}, \text{ADM}, \mathit{spk})$.
$\mathsf{SiProof}(\mathsf{sk},.,.,.)$**:** *On input* $(m, \sigma, \mathit{spk})$*, this oracle returns* $\mathit{SiProof}(\mathit{sk}, m, \sigma, \mathit{spk})$.
Let P *be a* SS*.* P *is* Immut *secure (or* immutable*) when for any polynomial time adversary* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *wins the following experiment is negligible, where* q_{Sig} *is the number of calls to the oracle* $\mathit{Sig}(.,\mathit{sk},.,.)$*,* $(m_i, \text{ADM}_i, \mathit{spk}_i)$ *is the* i^{th} *query asked to the oracle* $\mathit{Sig}(.,\mathit{sk},.,.)$ *and* σ_i *is the corresponding response:*

$\mathbf{Exp}^{\mathsf{Immut}}_{P,\mathcal{A}}(k)$*:*
$\mathit{set} \leftarrow \mathit{Init}(1^k)$
$(\mathit{pk}, \mathit{sk}) \leftarrow \mathit{SiGen}(\mathit{set})$
$(\mathit{spk}_*, m_*, \sigma_*) \leftarrow \mathcal{A}^{\mathit{Sig}(.,\mathit{sk},.,.),\mathit{SiProof}(\mathit{sk},.,.,.)}(\mathit{pk})$
if $(\mathit{Ver}(m_*, \sigma_*, \mathit{pk}, \mathit{spk}_*) = 1)$ *and* $(\forall\, i \in [\![1, q_{\mathit{Sig}}]\!],\ (\mathit{spk}_* \neq \mathit{spk}_i)$ *or*
$(\forall\ \text{MOD}$ *such that* $\text{ADM}_i(\text{MOD}) = 1, m_* \neq \text{MOD}(m_i)))$
then return 1*, else return* 0

Transparency: The transparency property guarantees that no adversary is able to distinguish whether a signature is sanitized or not. In addition to the signature oracle and the signer proof oracle, the adversary has access to a sanitize oracle $\mathsf{San}(.,.,.,.,\mathsf{ssk})$ that sanitizes chosen signatures and a sanitizer proof oracle $\mathsf{SaProof}(\mathsf{ssk},.,.,.)$ that computes sanitizer proofs for given signatures. Moreover the adversary has access to a challenge oracle $\mathsf{Sa/Si}(b,\mathsf{pk},\mathsf{spk},\mathsf{sk},\mathsf{ssk},.,.,.)$ that depends on a randomly chosen bit b: this oracle signs a given message and sanitizes it, if $b = 0$ then it outputs the original signature, otherwise it outputs the sanitized signature. The adversary cannot use the proof oracles on the signatures outputted by the challenge oracle. To succeed the experiment, the adversary must guess b.

Definition 12 (Transparency [8]). *Let the following oracles be:*
$\mathsf{San}(.,.,.,.,\mathsf{ssk})$**:** *On input* $(m, \text{MOD}, \sigma, \mathit{pk})$*, it returns* $\mathit{San}(m, \text{MOD}, \sigma, \mathit{pk}, \mathit{ssk})$.
$\mathsf{SaProof}(\mathsf{ssk},.,.,.)$**:** *On input* (m, σ, pk)*, this oracle returns* $\mathit{SaProof}(\mathit{ssk}, m, \sigma, \mathit{pk})$.
$\mathsf{Sa/Si}(b,\mathsf{pk},\mathsf{spk},\mathsf{sk},\mathsf{ssk},.,.,.)$**:** *On input* $(m, \text{ADM}, \text{MOD})$*, if* $\text{ADM}(\text{MOD}) = 0$*, this oracle returns* $\perp$*. Else if* $b = 0$*, this oracle returns* $\mathit{Sig}(\text{MOD}(m), \mathit{sk}, \mathit{spk}, \text{ADM})$*, else if* $b = 1$*, this oracle returns* $\mathit{San}(m, \text{MOD}, \mathit{Sig}(m, \mathit{sk}, \mathit{spk}, \text{ADM}), \mathit{pk}, \mathit{ssk})$.
Let P *be a* SS*.* P *is* Trans *secure (or* transparent*) when for any polynomial time adversary* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *wins the following experiment is negligible, where* $\mathit{Sig}(.,\mathit{sk},.,.)$ *and* $\mathit{SiProof}(\mathit{sk},.,.,.)$ *are defined as in Def. 11, and where* $S_{\mathit{Sa/Si}}$ *(resp.* S_{SiProof} *and* S_{SaProof}*) corresponds to the set of all signatures outputted by the oracle* $\mathit{Sa/Si}$ *(resp. sent to the oracles* $\mathit{SiProof}$ *and* $\mathit{SaProof}$*):*

$\mathbf{Exp}^{\mathsf{Trans}}_{P,\mathcal{A}}(k)$:
$\mathit{set} \leftarrow \mathit{Init}(1^k)$
$(\mathit{pk}, \mathit{sk}) \leftarrow \mathit{SiGen}(\mathit{set})$
$(\mathit{spk}, \mathit{ssk}) \leftarrow \mathit{SaGen}(\mathit{set})$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathit{Sig}(.,\mathit{sk},...),\mathit{San}(.,.,.,.,.,\mathit{ssk}),\mathit{SiProof}(\mathit{sk},.,.,.,.)}(\mathit{pk}, \mathit{spk})$
if $(b = b')$ *and* $(S_{\mathit{Sa/Si}} \cap (S_{\mathit{SiProof}} \cup S_{\mathit{SaProof}}) = \emptyset)$
then return 1, *else return* 0

Unlinkablility: The unlinkablility property ensures that a sanitized signature cannot be linked with the original one. We consider an adversary that has access to the signature oracle, the sanitize oracle, and both the signer and the sanitizer proof oracles. Moreover, the adversary has access to a challenge oracle $\mathsf{LRSan}(b, \mathsf{pk}, \mathsf{ssk}, ., .)$ that depends to a bit b: this oracle takes as input two signatures σ_0 and σ_1, the two corresponding messages m_0 and m_1 and two modification functions MOD_0 and MOD_1 chosen by the adversary. If the two signatures have the same admissible function ADM, if MOD_0 and MOD_1 are admissible according to ADM and if $\text{MOD}_0(m_0) = \text{MOD}_1(m_1)$ then the challenge oracle sanitizes σ_b using MOD_b and returns it. The goal of the adversary is to guess the bit b.

Definition 13 (Unlinkability [8]). *Let the following oracles be:*
$\mathsf{LRSan}(b, \mathsf{pk}, \mathsf{ssk}, ., .)$**:** *On input* $((m_0, \text{MOD}_0, \sigma_0)(m_1, \text{MOD}_1, \sigma_1))$*, if for* $i \in \{0,1\}$*,* $\mathit{Ver}(m_i, \sigma_i, \mathit{pk}, \mathit{spk}) = 1$ *and* $\text{ADM}_0 = \text{ADM}_1$ *and* $\text{ADM}_0(\text{MOD}_0) = 1$ *and* $\text{ADM}_1(\text{MOD}_1) = 1$ *and* $\text{MOD}_0(m_0) = \text{MOD}_1(m_1)$*, then this oracle returns* $\mathit{San}(m_b, \text{MOD}_b, \sigma_b, \mathit{pk}, \mathit{ssk})$*, else it returns* 0*.*
Let P *be a* SS *of security parameter* k*.* P *is* Unlink *secure (or* unlinkable*) when for any polynomial time adversary* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *wins the following experiment is negligibly close to* $1/2$*, where* $\mathit{Sig}(., \mathit{sk}, ., .)$ *and* $\mathit{SiProof}(\mathit{sk}, ., ., .)$ *are defined as in Def. 11 and* $\mathit{San}(., ., ., ., \mathit{ssk})$ *and* $\mathit{SaProof}(\mathit{ssk}, ., ., .)$ *are defined as in Def. 12:*

$\mathbf{Exp}^{\mathsf{Unlink}}_{P,\mathcal{A}}(k)$:
$\mathit{set} \leftarrow \mathit{Init}(1^k)$
$(\mathit{pk}, \mathit{sk}) \leftarrow \mathit{SiGen}(\mathit{set})$
$(\mathit{spk}, \mathit{ssk}) \leftarrow \mathit{SaGen}(\mathit{set})$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathit{SiProof}(\mathit{sk},.,.,.),\mathit{SaProof}(\mathit{ssk},.,.,.),\mathit{LRSan}(b,\mathit{pk},\mathit{spk},.,.)}(\mathit{pk}, \mathit{spk})$
if $(b = b')$ *then return* 1, *else return* 0

Accountability: Standard defintion of accountability is shared into two security experiments: the sanitizer accountability and the signer accountability. In the sanitizer accountability experiment, the adversary has access to the signature oracle and the signer proof oracle. Its goal is to forge a signature such that the signer proof algorithm returns a proof that this signature is not sanitized. To succeed the experiment, this signature must not come from the signature oracle.

Definition 14 (Sanitizer Accountability [8]). *Let P be a SS. P is SaAcc-1 secure (or* sanitizer accountable*) when for any polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following experiment is negligible, where the oracles $\mathsf{Sig}(.,\mathsf{sk},.,.)$ and $\mathsf{SiProof}(\mathsf{sk},.,.,.)$ are defined as in Def. 11, q_{Sig} is the number of calls to the oracle $\mathsf{Sig}(.,\mathsf{sk},.,.)$, the tuple $(m_i, \text{ADM}_i, \mathsf{spk}_i)$ is the i^{th} query asked to the oracle $\mathsf{Sig}(.,\mathsf{sk},.,.)$ and σ_i is the corresponding response:*

$\mathbf{Exp}^{\mathsf{SaAcc\text{-}1}}_{P,\mathcal{A}}(k)$*:*
$\mathsf{set} \leftarrow \mathsf{Init}(1^k)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SiGen}(\mathsf{set})$
$(\mathsf{spk}_*, m_*, \sigma_*) \leftarrow \mathcal{A}^{\mathsf{Sig}(.,\mathsf{sk},.,.),\mathsf{SiProof}(\mathsf{sk},.,.,.)}(\mathsf{pk})$
$\pi^*_{si} \leftarrow \mathsf{SiProof}(\mathsf{sk}, m_*, \sigma_*, \mathsf{spk}_*)$
if $\forall\ i \in [\![1, q_{\mathsf{Sig}}]\!]$, $(\sigma_* \neq \sigma_i)$
  and $(\mathsf{Ver}(m_*, \sigma_*, \mathsf{pk}, \mathsf{spk}_*) = 1)$
  and $(\mathsf{SiJudge}(m_*, \sigma_*, \mathsf{pk}, \mathsf{spk}_*, \pi^*_{si}) \neq 0)$
then return 1*, else return* 0

In the signer accountability experiment, the adversary knows the public key of the sanitizer and has access to the sanitize oracle and the sanitizer proof oracle. Its goal is to forge a signature together with a proof that this signature is sanitized. To succeed the experiment, this signature must not come from the sanitize oracle.

Definition 15 (Signer Accountability [8]). *Let P be a SS. P is SiAcc-1 secure (or* signer accountable*) when for any polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following experiment is negligible, where the oracle $\mathsf{San}(.,.,.,.,\mathsf{ssk})$ and $\mathsf{SaProof}(\mathsf{ssk},.,.,.)$ are defined as in Def. 12 and where q_{San} is the number of calls to the oracle $\mathsf{San}(.,.,.,.,\mathsf{ssk})$, $(m_i, \text{MOD}_i, \sigma_i, \mathsf{pk}_i)$ is the i^{th} query asked to the oracle $\mathsf{San}(.,.,.,.,\mathsf{ssk})$ and σ'_i is the corresponding response:*

$\mathbf{Exp}^{\mathsf{SiAcc\text{-}1}}_{P,\mathcal{A}}(k)$*:*
$\mathsf{set} \leftarrow \mathsf{Init}(1^k)$
$(\mathsf{spk}, \mathsf{ssk}) \leftarrow \mathsf{SaGen}(\mathsf{set})$
$(\mathsf{pk}_*, m_*, \sigma_*, \pi^*_{si}) \leftarrow \mathcal{A}^{\mathsf{San}(.,.,.,.,\mathsf{ssk}),\mathsf{SaProof}(\mathsf{ssk},.,.,.)}(\mathsf{spk})$
if $\forall\ i \in [\![1, q_{\mathsf{San}}]\!]$, $(\sigma_* \neq \sigma'_i)$
  and $(\mathsf{Ver}(m_*, \sigma_*, \mathsf{pk}_*, \mathsf{spk}) = 1)$
  and $(\mathsf{SiJudge}(m_*, \sigma_*, \mathsf{pk}_*, \mathsf{spk}, \pi^*_{si}) = 0)$
then return 1*, else return* 0

Strong Accountability: Since our definition of sanitizable signature provides a second proof algorithm for the sanitizer, we define two additional security experiments (for signer and sanitizer accountability) to ensure the soundness of the proofs computed by this algorithm. We say that a scheme is strongly accountable when it is signer and sanitizer accountable for both the signer and the sanitizer proof algorithms.

Thus, in our second signer accountability experiment, we consider an adversary that has access to the sanitize oracle and the sanitizer proof oracle. Its goal is to forge a signature such that the sanitizer proof algorithm returns a proof that this signature is sanitized. To win the experiment, this signature must not come from the sanitize oracle.

Definition 16 (Strong Signer Accountability). *Let P be a SS. P is SiAcc-2 secure when for any polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following experiment is negligible, where q_{San} is the number of calls to the oracle $\mathsf{San}(.,.,.,.,\mathsf{ssk})$, $(m_i, \text{MOD}_i, \sigma_i, \mathsf{pk}_i)$ is the i^{th} query asked to the oracle $\mathsf{San}(.,.,.,.,\mathsf{ssk})$ and σ'_i is the corresponding response:*

$\mathbf{Exp}^{\mathsf{SiAcc\text{-}2}}_{P,\mathcal{A}}(k)$:
$\mathsf{set} \leftarrow \mathsf{Init}(1^k)$
$(\mathsf{spk}, \mathsf{ssk}) \leftarrow \mathsf{SaGen}(\mathsf{set})$
$(\mathsf{pk}_*, m_*, \sigma_*) \leftarrow \mathcal{A}^{\mathsf{San}(.,.,.,.,\mathsf{ssk}),\mathsf{SaProof}(\mathsf{ssk},.,.,.)}(\mathsf{spk})$
$\pi^*_{sa} \leftarrow \mathsf{SaProof}(\mathsf{ssk}, m_*, \sigma_*, \mathsf{pk}_*)$
if $\forall\ i \in [\![1, q_{San}]\!]$, $(\sigma_* \neq \sigma'_i)$
  and $(\mathsf{Ver}(m_*, \sigma_*, \mathsf{pk}_*, \mathsf{spk}) = 1)$
  and $(\mathsf{SaJudge}(m_*, \sigma_*, \mathsf{pk}_*, \mathsf{spk}, \pi^*_{sa}) \neq 1)$
then return 1, *else return* 0

P *is* strong signer accountable *when it is both* SiAcc-1 *and* SiAcc-2 *secure.*

Finally, in our second sanitizer accountability experiment, we consider an adversary that knows the public key of the signer and has access to the signer oracle and the signer proof oracle. Its goal is to sanitize a signature with a proof that this signature is not sanitized. To win the experiment, this signature must not come from the signer oracle.

Definition 17 (Strong Sanitizer Accountability). *Let P be a SS. P is SaAcc-2 secure when for any polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following experiment is negligible, $\mathsf{Sig}(.,\mathsf{sk},.,.)$ and $\mathsf{SiProof}(\mathsf{sk},.,.,.)$ are defined as in Def. 11, q_{Sig} is the number of calls to the oracle $\mathsf{Sig}(.,\mathsf{sk},.,.)$, $(m_i, \text{ADM}_i, \mathsf{spk}_i)$ is the i^{th} query asked to the oracle $\mathsf{Sig}(.,\mathsf{sk},.,.)$ and σ_i is the corresponding response:*

$\mathbf{Exp}^{\mathsf{SaAcc\text{-}2}}_{P,\mathcal{A}}(k)$:
$\mathsf{set} \leftarrow \mathsf{Init}(1^k)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SaGen}(\mathsf{set})$
$(\mathsf{spk}_*, m_*, \sigma_*, \pi^*_{sa}) \leftarrow \mathcal{A}^{\mathsf{Sig}(.,\mathsf{sk},.,.),\mathsf{SiProof}(\mathsf{sk},.,.,.)}(\mathsf{spk})$
if $\forall\ i \in [\![1, q_{Sig}]\!]$, $(\sigma_* \neq \sigma_i)$
  and $(\mathsf{Ver}(m_*, \sigma_*, \mathsf{pk}, \mathsf{spk}_*) = 1)$
  and $(\mathsf{SaJudge}(m_*, \sigma_*, \mathsf{pk}, \mathsf{spk}_*, \pi^*_{sa}) = 1)$
then return 1, *else return* 0

P *is* strong sanitizer accountable *when it is both* SaAcc-1 *and* SaAcc-2 *secure.*

5 An Efficient Verifiable Ring Signature: EVeR

We present our VRS scheme called EVeR (for *Efficient VErifiable Ring signature*). EVeR works as follows: the signer produces an anonymous commitment from his secret key and the message (*i.e.* a commitment that leaks no information about the secret key), then he proves that this commitment was produced with a secret key corresponding to one of the public keys of the group members using a zero-knowledge proof system. Note that the same methodology was used to design several ring signature schemes of the literature [2, 15, 17, 24]. Moreover, to prove that he is (resp. he is not) the signer of

a message, the user proves that the commitment was (resp. was not) produced from the secret key that corresponds to his public key using a zero-knowledge proof system. Our scheme is based on the DDH assumption and uses a NIZKP of equality of two discrete logarithms out of n elements. We show how to build this NIZKP: Let $\mathbb{G}$ be a group of prime order p, n be an integer and let the following binary relation be:

$$\mathcal{R}_n = \left\{ \begin{array}{c} s = \{(h_i, z_i, g_i, y_i)\}_{1 \leq i \leq n}; \\ (s, w) : \exists i \in [\![1, n]\!], (((h_i, z_i, g_i, y_i) \in \mathbb{G}^4) \\ \wedge (w = \log_{g_i}(y_i) = \log_{h_i}(z_i))); \end{array} \right\}.$$

We denote by $\mathcal{L}_n$ the language $\{s : \exists w, (s, w) \in \mathcal{R}_n\}$. Consider the case $n = 1$. In [18], authors present an interactive zero-knowledge proof of knowledge system for the relation $\mathcal{R}_1$. It proves the equality of two discrete logarithms. For example using $(h, z, g, y) \in \mathcal{L}_1$, a prover convinces a verifier that $\log_g(y) = \log_h(z)$. The witness used by the prover is $w = \log_g(y)$. This proof system is a *sigma protocol* in the sense that there are only three interactions: the prover sends a commitment, the verifier sends a challenge, and the prover returns a response.

To transform the proof system of $\mathcal{R}_1$ into a generic proof system of any $\mathcal{R}_n$, we use the generic transformation given in [19]. For any integer n and any relation $\mathcal{R}$, the authors show how to transform a zero-knowledge proof of knowledge of a witness w such that $(s, w) \in \mathcal{R}$ for a given statement s into a zero-knowledge proof of knowledge of a witness w such that there exists $s \in S$ such that $(s, w) \in \mathcal{R}$ for a given set of n statements S, under the condition that the proof is a sigma protocol. Note that the resulting proof system is also a sigma protocol.

The final step is to transform it into a non-interactive proof system. We use the well-known Fiat-Shamir transformation [20]. This transformation changes any interactive proof system that is a sigma protocol into a non interactive one. The resulting proof system is complete, sound and zero-knowledge in the random oracle model. Finally, we obtain the following scheme.

Scheme 2 (LogEq_n) *Let $\mathbb{G}$ be a group of prime order p, $H : \{0,1\} \to \mathbb{Z}_p^*$ be a hash function and n be an integer. We define the NIZKP $\mathsf{LogEq}_n = (\mathsf{LEprove}_n, \mathsf{LEverif}_n)$ for $\mathcal{R}_n$ by:*

$\mathsf{LEprove}_n(\{(h_i, z_i, g_i, y_i)\}_{1 \leq i \leq n}, x)$. *Let $x = \log_{g_j}(y_j) = \log_{h_j}(z_j)$, this algorithm picks $r_j \xleftarrow{\$} \mathbb{Z}_p^*$, computes $R_j = g_j^{r_j}$ and $S_j = h_j^{r_j}$. For all $i \in [\![1, n]\!]$ and $i \neq j$, it picks $c_i \xleftarrow{\$} \mathbb{Z}_p^*$ and $\gamma_i \xleftarrow{\$} \mathbb{Z}_p^*$, and computes $R_i = g_i^{\gamma_i}/y_i^{c_i}$ and $S_i = h_i^{\gamma_i}/z_i^{c_i}$. It computes $c = H(R_1||S_1||\ldots||R_n||S_n)$. It then computes $c_j = c/(\prod_{i=1;i\neq j}^{n} c_i)$ and $\gamma_j = r_j + c_j \cdot x$. It outputs $\pi = \{(R_i, S_i, c_i, \gamma_i)\}_{1 \leq i \leq n}$.*

$\mathsf{LEverif}_n(\{(h_i, z_i, g_i, y_i)\}_{1 \leq i \leq n}, \pi)$. *It parses $\pi = \{(R_i, S_i, c_i, \gamma_i)\}_{1 \leq i \leq n}$. If $H(R_1||S_1||\ldots||R_n||S_n) \neq \prod_{i=1;i\neq j}^{n} c_i$ then it returns 0. Else if there exists $i \in [\![1, n]\!]$ such that $g_i^{\gamma_i} \neq R_i \cdot y_i^{c_i}$ or $h_i^{\gamma_i} \neq S_i \cdot z_i^{c_i}$ then it returns 0, else 1.*

Theorem 1. *The NIZKP LogEq_n is a proof of knowledge, moreover it is complete, sound, and zero-knowledge in the random oracle model.*

The proof of this theorem is a direct implication of [18], [19] and [20]. Using this proof system, we build our VRS scheme called EVeR:

Scheme 3 (Efficient Verifiable Ring Signature (EVeR)) EVeR *is a VRS defined by:*

V.Init(1^k): *It generates a prime order group setup* $(\mathbb{G}, p, g)$ *and a hash function* $H : \{0,1\}^* \to \mathbb{G}$. *It returns the setup* $\mathit{set} = (\mathbb{G}, p, g, H)$.

V.Gen(set): *It picks* $\mathit{sk} \xleftarrow{\$} \mathbb{Z}_p^*$, *computes* $\mathit{pk} = g^{\mathit{sk}}$ *and returns a pair of signer public/private keys* $(\mathit{pk}, \mathit{sk})$.

V.Sig(L, m, sk): *It picks* $r \xleftarrow{\$} \mathbb{Z}_p^*$, *it computes* $h = H(m||r)$ *and* $z = h^{\mathit{sk}}$, *it runs* $P \leftarrow \mathsf{LEprove}_{|L|}(\{(h, z, g, \mathit{pk}_l)\}_{\mathit{pk}_l \in L}, \mathit{sk})$ *and returns* $\sigma = (r, z, P)$.

V.Ver(L, m, σ): *It parses* $\sigma = (r, z, P)$, *computes* $h = H(m||r)$ *and returns* $b \leftarrow \mathsf{LEverif}_{|L|}(\{(h, z, g, \mathit{pk}_l)\}_{\mathit{pk}_l \in L}, P)$.

V.Proof$(L, m, \sigma, \mathsf{pk}, \mathsf{sk})$: *It parses* $\sigma = (r, z, P)$, *computes* $h = H(m||r)$ *and* $\bar{z} = h^{\mathit{sk}}$, *runs* $\bar{P} \leftarrow \mathsf{LEprove}_1(\{(h, \bar{z}, g, \mathit{pk})\}, \mathit{sk})$ *and returns* $\pi = (\bar{z}, \bar{P})$.

V.Judge$(L, m, \sigma, \mathsf{pk}, \pi)$: *It parses* $\sigma = (r, z, P)$ *and* $\pi = (\bar{z}, \bar{P})$, *computes* $h = H(m||r)$ *and runs* $b \leftarrow \mathsf{LEverif}_1(\{(h, \bar{z}, g, \mathit{pk})\}, \pi)$. *If* $b \neq 1$ *then it returns* $\perp$. *Else, if* $z = \bar{z}$ *then it returns* 1, *else it returns* 0.

Theorem 2. EVeR *is unforgeable, anonymous, accountable and non-seizable under the DDH assumption in the random oracle model.*

We give the intuition of these properties, the proof of the theorem is given in the full version of this paper [12]:

Unforgeability: The scheme is unforgeable since nobody can prove that $\log_g(\mathsf{pk}_l) = \log_h(z)$ without the knowledge of $\mathsf{sk} = \log_h(z)$.

Anonymity: Breaking the anonymity of such a signature is equivalent to breaking the DDH assumption. Indeed, to link a signature $z = h^{\mathsf{sk}}$ with the corresponding public key of Alice $\mathsf{pk} = g^{\mathsf{sk}}$, an attacker must solve the DDH problem on the instance (pk, h, z). Moreover, note that since the value r *randomizes* the signature, it is not possible to link two signatures of the same message produced by Alice.

Accountability: To break the accountability, an adversary must forge a valid signature (*i.e.* to prove that there exists pk_l in the group such that $\log_g(\mathsf{pk}_l) \neq \log_h(z)$) and to prove that he is not the signer (*i.e.* $\log_g(\mathsf{pk}) \neq \log_h(z)$ where pk is the public key chosen by the adversary). However, since the adversary does not know the secret keys of the other members of the group, he would have to break the soundness of LogEq to win the experiment, which is not possible.

Non-seizable: (non-sei-1) no adversary is able to forge a proof that it is the signer of a signature produced by another user since it is equivalent to proving a false statement using a sound NIZKP. (non-sei-2) the proof algorithm run by a honest user with the public key pk returns a proof that this user is the signer of a given signature only if $\log_g(\mathsf{pk}) = \log_h(z)$. Since no adversary is able to compute z such that $\log_g(\mathsf{pk}) = \log_h(z)$ without the corresponding secret key, no adversary is able to break the non-seizability of EvER.

6 Our Unlinkable Sanitizable Signature Scheme: GUSS

We present our USS instantiated by a digital signature (DS) scheme and a VRS.

Scheme 4 (Generic Unlinkable Sanitizable Signature (GUSS)) *Let D be a deterministic digital signature scheme such that $D = (\mathsf{D.Init}, \mathsf{D.Gen}, \mathsf{D.Sig}, \mathsf{D.Ver})$ and V be a verifiable ring signature scheme such that $V = (\mathsf{V.Init}, \mathsf{V.Gen}, \mathsf{V.Sig}, \mathsf{V.Ver}, \mathsf{V.Proof}, \mathsf{V.Judge})$.* GUSS *instantiated with (D, V) is a sanitizable signature scheme defined by the following algorithms:*

$\mathsf{Init}(1^k)$: *It runs $\mathsf{set}_d \leftarrow \mathsf{D.Init}(1^k)$ and $\mathsf{set}_v \leftarrow \mathsf{V.Init}(1^k)$. Then it returns the setup $\mathsf{set} = (\mathsf{set}_d, \mathsf{set}_v)$.*

$\mathsf{SiGen}(\mathsf{set})$: *It parses $\mathsf{set} = (\mathsf{set}_d, \mathsf{set}_v)$, runs $(\mathsf{pk}_d, \mathsf{sk}_d) \leftarrow \mathsf{D.Gen}(\mathsf{set}_d)$ and $(\mathsf{pk}_v, \mathsf{sk}_v) \leftarrow \mathsf{V.Gen}(\mathsf{set}_v)$. Then it returns $(\mathsf{pk}, \mathsf{sk})$ where $\mathsf{pk} = (\mathsf{pk}_d, \mathsf{pk}_v)$ and $\mathsf{sk} = (\mathsf{sk}_d, \mathsf{sk}_v)$.*

$\mathsf{SaGen}(\mathsf{set})$: *It parses $\mathsf{set} = (\mathsf{set}_d, \mathsf{set}_v)$ and runs $(\mathsf{spk}, \mathsf{ssk}) \leftarrow \mathsf{V.Gen}(\mathsf{set}_v)$. It returns $(\mathsf{spk}, \mathsf{ssk})$.*

$\mathsf{Sig}(m, \mathsf{sk}, \mathsf{spk}, \mathrm{ADM})$: *It parses $\mathsf{sk} = (\mathsf{sk}_d, \mathsf{sk}_v)$. It first computes the fixed message part $M \leftarrow \mathrm{FIX}_{\mathrm{ADM}}(m)$ and runs $\sigma_1 \leftarrow \mathsf{D.Sig}(\mathsf{sk}_d, (M||\mathrm{ADM}||\mathsf{pk}||\mathsf{spk}))$ and $\sigma_2 \leftarrow \mathsf{V.Sig}(\{\mathsf{pk}_v, \mathsf{spk}\}, \mathsf{sk}_v, (\sigma_1||m)))$. It returns $\sigma = (\sigma_1, \sigma_2, \mathrm{ADM})$.*

$\mathsf{San}(m, \mathrm{MOD}, \sigma, \mathsf{pk}, \mathsf{ssk})$: *It parses $\sigma = (\sigma_1, \sigma_2, \mathrm{ADM})$ and $\mathsf{pk} = (\mathsf{pk}_d, \mathsf{pk}_v)$. This algorithm first computes the modified message $m' \leftarrow \mathrm{MOD}(m)$ and it runs $\sigma_2' \leftarrow \mathsf{V.Sig}(\{\mathsf{pk}_v, \mathsf{spk}\}, \mathsf{ssk}, (\sigma_1||m'))$. It returns $\sigma' = (\sigma_1, \sigma_2', \mathrm{ADM})$.*

$\mathsf{Ver}(m, \sigma, \mathsf{pk}, \mathsf{spk})$: *It parses $\sigma = (\sigma_1, \sigma_2, \mathrm{ADM})$ and it computes the fixed message part $M \leftarrow \mathrm{FIX}_{\mathrm{ADM}}(m)$. Then it runs $b_1 \leftarrow \mathsf{D.Ver}(\mathsf{pk}_d, (M||\mathrm{ADM}||\mathsf{pk}||\mathsf{spk}), \sigma_1)$ and $b_2 \leftarrow \mathsf{V.Ver}(\{\mathsf{pk}_d, \mathsf{spk}\}, (\sigma_1||m), \sigma_2)$. It returns $b = (b_1 \wedge b_2)$.*

$\mathsf{SiProof}(\mathsf{sk}, m, \sigma, \mathsf{spk})$: *It parses $\sigma = (\sigma_1, \sigma_2, \mathrm{ADM})$ and the key $\mathsf{sk} = (\mathsf{sk}_d, \mathsf{sk}_v)$. It runs $\pi_{\mathsf{si}} \leftarrow \mathsf{V.Proof}(\{\mathsf{pk}_v, \mathsf{spk}\}, (m||\sigma_1), \sigma_2, \mathsf{pk}_v, \mathsf{sk}_v)$ and returns it.*

$\mathsf{SaProof}(\mathsf{ssk}, m, \sigma, \mathsf{pk})$: *It parses the signature $\sigma = (\sigma_1, \sigma_2, \mathrm{ADM})$. It runs $\pi_{\mathsf{sa}} \leftarrow \mathsf{V.Proof}(\{\mathsf{pk}_v, \mathsf{spk}\}, (m||\sigma_1), \sigma_2, \mathsf{spk}, \mathsf{ssk})$ and returns it.*

$\mathsf{SiJudge}(m, \sigma, \mathsf{pk}, \mathsf{spk}, \pi_{\mathsf{si}})$: *It parses $\sigma = (\sigma_1, \sigma_2, \mathrm{ADM})$ and $\mathsf{pk} = (\mathsf{pk}_d, \mathsf{pk}_v)$. It runs $b \leftarrow \mathsf{V.Judge}(\{\mathsf{pk}_v, \mathsf{spk}\}, (m||\sigma_1), \sigma_2, \mathsf{pk}_v, \pi_{\mathsf{si}})$ and returns it.*

$\mathsf{SaJudge}(m, \sigma, \mathsf{pk}, \mathsf{spk}, \pi_{\mathsf{sa}})$: *It parses $\sigma = (\sigma_1, \sigma_2, \mathrm{ADM})$ and $\mathsf{pk} = (\mathsf{pk}_d, \mathsf{pk}_v)$. It runs $b \leftarrow \mathsf{V.Judge}(\{\mathsf{pk}_v, \mathsf{spk}\}, (m||\sigma_1), \sigma_2, \mathsf{spk}, \pi_{\mathsf{sa}})$ and returns $(1 - b)$.*

The signer secret key $\mathsf{sk} = (\mathsf{sk}_d, \mathsf{sk}_v)$ contains a secret key sk_d compatible with the DS scheme and a secret key sk_v compatible with the VRS scheme. The signer public key $\mathsf{pk} = (\mathsf{pk}_d, \mathsf{pk}_v)$ contains the two corresponding public keys. The sanitizer public/secret key pair $(\mathsf{spk}, \mathsf{ssk})$ is generated as in the VRS scheme.

Let m be a message and M be the *fixed part* chosen by the signer according to the admissible function ADM. To sign m, the signer first signs M together with the public key of the sanitizer spk and the admissible function ADM using the DS scheme. We denote this signature by σ_1. The signer then signs in σ_2 the full message m together with σ_1 using the VRS scheme for the set of public keys $L = \{\mathsf{pk}_v, \mathsf{spk}\}$. Informally, he anonymously signs $(\sigma_1||m)$ within a group of two users: the signer and the sanitizer. The final sanitizable signature is $\sigma = (\sigma_1, \sigma_2)$. The verification algorithm works in two steps: it verifies the signature σ_1 and it verifies the anonymous signature σ_2.

To sanitize this signature $\sigma = (\sigma_1, \sigma_2)$, the sanitizer chooses an admissible message m' according to ADM (*i.e.* m and m' have the same fixed part). Then he anonymously signs m' together with σ_1 using the VRS for the group $L = \{\mathsf{pk}_v, \mathsf{spk}\}$ using the secret key ssk. We denote by σ'_2 this signature. The final sanitized signature is $\sigma' = (\sigma_1, \sigma'_2)$.

Theorem 3. *For any deterministic and unforgeable DS scheme D and any unforgeable, anonymous, accountable and non-seizable VRS scheme V,* GUSS *instantiated with (D, V) is immutable, transparent, strongly accountable and unlinkable.*

We give the intuition of these properties, the proof of the theorem is given in the full version of this paper [12]:

Transparency: According to the anonymity of σ_2 and σ'_2, nobody can guess if a signature comes from the signer or the sanitizer, and since both signatures have the same structure, nobody can guess whether a signature is sanitized or not.

Immutability: Since it is produced by a unforgeable DS scheme, nobody can forge the signature σ_1 of the fixed part M without the signer secret key. Thus the sanitizer cannot change the fixed part of the signatures. Moreover, since σ_1 signs the public key of the sanitizer in addition to M, the other users cannot forge a signature of an admissible message using σ_1.

Unlinkability: An adversary knows (i) two signatures σ^0 and σ^1 that have the same fixed part M according to the same function ADM for the same sanitizer and (ii) the sanitized signature $\sigma' = (\sigma'_1, \sigma'_2)$ computed from σ^b for a given admissible message m' and an unknown bit b. To achieve unlinkability, it must be hard to guess b. Since the DS scheme is deterministic, the two signatures $\sigma^0 = (\sigma^0_1, \sigma^0_2)$ and $\sigma^1 = (\sigma^1_1, \sigma^1_2)$ have the same first part (*i.e.* $\sigma^0_1 = \sigma^1_1$). As it was shown before, the σ' has the same first part σ'_1 as the original one, thus $\sigma'_1 = \sigma^0_1 = \sigma^1_1$ and σ'_1 leaks no information about b. On the other hand, the second part of the sanitized signature σ'_2 is computed from the modified message m' and the first part of the original signature. Since $\sigma^0_1 = \sigma^1_1$, we deduce that σ'_2 leaks no information about b. Finally, the best strategy of the adversary is to randomly guess b.

(Strong) Accountability: the signer must be able to prove the provenance of a signature. It is equivalent to breaking the anonymity of the second parts σ_2 of this signature: if it was created by the signer then it is the original signature, else it was created by the sanitizer and it is a sanitized signature. By definition, the VRS scheme used to generate σ_2 provides a way to prove whether a user is the author of a signature or not. GUSS uses it in its proof algorithm to achieve accountability. Note that since the sanitizer uses the same VRS scheme to sanitize a signature, he also can prove the origin of a given signature to achieve the strong accountability.

7 Conclusion

In this paper, we revisit the notion of verifiable ring signatures. We improve its properties of verifiability, we give a security model for this primitive and we design a simple, efficient and secure scheme named EvER. We extend the security model of sanitizable signatures in order to allow the sanitizer to prove the origin of a signature. Finally, we design a generic unlinkable sanitizable signature scheme named GUSS based on

verifiable ring signatures. This scheme is twice as efficient as the best scheme of the literature. In the future, we aim at finding other applications for the verifiable ring signatures that require our security properties.

8 Acknowledgement

We acknowledge Dominique Schröder for his helpful comments and feedback on our paper.

References

1. Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. *Sanitizable Signatures*, pages 159–177. Springer Berlin Heidelberg, 2005.
2. Man Ho Au, Willy Susilo, and Siu-Ming Yiu. *Event-Oriented k-Times Revocable-iff-Linked Group Signatures*, pages 223–234. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
3. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Brickell [6], pages 390–420.
4. Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 60–79. Springer, March 2006.
5. Dan Boneh. The decision Diffie-Hellman problem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *LNCS*. Springer, 1998. Invited paper.
6. Ernest F. Brickell, editor. *CRYPTO'92*, volume 740 of *LNCS*. Springer, August 1993.
7. Christina Brzuska, Heike Busch, Oezguer Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, and Dominique Schröder. *Redactable Signatures for Tree-Structured Data: Definitions and Constructions*. Springer Berlin Heidelberg, 2010.
8. 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*, volume 5443 of *LNCS*, pages 317–336. Springer, March 2009.
9. Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder. Unlinkability of sanitizable signatures. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 444–461. Springer, May 2010.
10. Christina Brzuska, Henrich C. Pöhls, and Kai Samelin. *Non-interactive Public Accountability for Sanitizable Signatures*, pages 178–193. Berlin, Heidelberg, 2013.
11. Christina Brzuska, Henrich C. Pöhls, and Kai Samelin. *Efficient and Perfectly Unlinkable Sanitizable Signatures without Group Signatures*, pages 12–30. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.
12. Xavier Bultel and Pascal Lafourcade. Unlinkable and strongly accountable sanitizable signatures from verifiable ring signatures. Cryptology ePrint Archive, Report 2017/605, 2017. `http://eprint.iacr.org/2017/605`.
13. Sébastien Canard and Amandine Jambert. *On Extended Sanitizable Signature Schemes*, pages 179–194. Springer Berlin Heidelberg, 2010.
14. Sébastien Canard, Amandine Jambert, and Roch Lescuyer. *Sanitizable Signatures with Several Signers and Sanitizers*, pages 35–52. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

15. Sbastien Canard, Berry Schoenmakers, Martijn Stam, and Jacques Traor. List signature schemes. *Discrete Applied Mathematics*, 154(2):189 – 201, 2006.
16. Z. Changlun, L. Yun, and H. Dequan. A new verifiable ring signature scheme based on nyberg-rueppel scheme. In *2006 8th international Conference on Signal Processing*, volume 4, 2006.
17. Melissa Chase and Anna Lysyanskaya. *On Signatures of Knowledge*, pages 78–96. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
18. David Chaum and Torben P. Pedersen. Wallet databases with observers. In Brickell [6], pages 89–105.
19. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO94*, volume 839 of *LNCS*. Springer, 1994.
20. Amos Fiat and Adi Shamir. *Advances in Cryptology — CRYPTO' 86: Proceedings*, chapter How To Prove Yourself: Practical Solutions to Identification and Signature Problems, pages 186–194. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987.
21. N. Fleischhacker, J. Krupp, G. Malavolta, J. Schneider, D. Schrder, and M. Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In *Public-Key Cryptography PKC 2016*, LNCS. Springer, 2016.
22. Georg Fuchsbauer and David Pointcheval. *Anonymous Proxy Signatures*. Springer Berlin Heidelberg, 2008.
23. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
24. Fumitaka Hoshino, Tetsutaro Kobayashi, and Koutarou Suzuki. *Anonymizable Signature and Its Construction from Pairings*, pages 62–77. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
25. Robert Johnson, David Molnar, Dawn Song, and David Wagner. *Homomorphic Signature Schemes*, pages 244–262. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.
26. Russell W. F. Lai, Tao Zhang, Sherman S. M. Chow, and Dominique Schröder. *Efficient Sanitizable Signatures Without Random Oracles*, pages 363–380. Springer International Publishing, 2016.
27. K. C. Lee, H. A. Wen, and T. Hwang. Convertible ring signature. *IEE Proceedings - Communications*, 152(4):411–414, 2005.
28. Jiqiang Lv and Xinmei Wang. *Verifiable Ring Signature*, pages 663–665. DMS Proceedings, 2003.
29. Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, December 2001.
30. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, August 1990.
31. Ron Steinfeld, Laurence Bull, and Yuliang Zheng. *Content Extraction Signatures*, pages 285–304. Springer Berlin Heidelberg, 2002.
32. Shangping Wang, Rui Ma, Yaling Zhang, and Xiaofeng Wang. Ring signature scheme based on multivariate public key cryptosystems. *Computers and Mathematics with Applications*, 62(10):3973 – 3979, 2011.

A Algorithms Complexity

In this section, we detail the complexity of the algorithms of our schemes. More precisely, we give the number of exponentiations in a prime order group for each algorithm. Since our schemes are pairing free, this is the main operation. Moreover, we give the size of some values outputted by these algorithms (keys, signatures and proofs). This

size is given in the number of elements of a group of prime order p. For the sake of clarity, we do not distinguish between elements of a group $\mathbb{G}$ of prime order p where the DDH assumption is hard and elements of $\mathbb{Z}_p^*$.

In Figure 2, we give the number of exponentiations of each algorithm of Schnorr's signature, and we give the size of the secret/public keys $\mathsf{sk}_{\mathsf{Sh}}$ and $\mathsf{pk}_{\mathsf{Sh}}$ and the size of a signature σ_{Sh}.

Schnorr	D.Gen	D.Sig	D.Ver	$\mathsf{sk}_{\mathsf{Sh}}$	$\mathsf{pk}_{\mathsf{Sh}}$	σ_{Sh}
exp/size	1	1	2	1	1	2

Fig. 2. Complexity analysis of Schnorr (Scheme 1).

In Figure 3, we give the number of exponentiations of each algorithm of the LogEq_n proof system and the size of a proof π_n^{LE} depending to the number n. The first line corresponds to the general case, the two other lines correspond to the case where $n = 1$ and $n = 2$.

LogEq_n	$\mathsf{LEprove}_n$	$\mathsf{LEverif}_n$	π_n^{LE}
n	$2 + 4 \cdot (n-1)$	$4 \cdot n$	$4 \cdot n$
$n = 1$	2	4	4
$n = 2$	6	8	8

Fig. 3. Complexity analysis of LogEq (Scheme 2).

EVeR	V.Gen	$\mathsf{V.Sig}_n$	$\mathsf{V.Ver}_n$	V.Proof	V.Judge
n (generic)	1	$1 + \mathsf{LEprove}_n$	$\mathsf{LEverif}_n$	$1 + \mathsf{LEprove}_1$	$\mathsf{LEverif}_1$
$n = 2$ (with LogEq_n)	1	7	8	3	4

EVeR	$\mathsf{sk}_{\mathrm{EV}}$	$\mathsf{pk}_{\mathrm{EV}}$	σ_n^{EV}	π_n^{EV}
n (generic)	1	1	$2 + \pi_n^{\mathsf{LE}}$	$1 + \pi_1^{\mathsf{LE}}$
$n = 2$ (with LogEq_n)	1	1	10	5

Fig. 4. Complexity analysis of EVeR (Scheme 3).

In Figure 4, we give the number of exponentiations of each algorithm of the EVeR verifiable ring signature scheme (first table) and the size the secret/public keys $\mathsf{sk}_{\mathrm{EV}}$ and $\mathsf{pk}_{\mathrm{EV}}$, the size of a signature σ_n^{EV} and the size of a proof π_n^{EV} (second table). These values depend on the size of the ring n. The first line corresponds to the generic case, where the values depend on the chosen proof system. The second line corresponds to the case where $n = 2$ and where the proof system is LogEq_2.

In Figure 5, we give the number of exponentiations of each algorithm of the GUSS verifiable ring signature scheme (first table) and the size of the secret/public keys of

GUSS	SiGen	SaGen	Sig	San	Ver	SiProof
generic	D.Gen + V.Gen	V.Gen	D.Sig + V.Sig_2	V.Sig_2	D.Ver + V.Ver_2	V.Proof
EvER and Schnorr	2	1	8	7	10	3

GUSS	SiJudge	sk	pk	ssk	spk	σ	π
generic	V.Judge	$\mathsf{sk}_{\mathrm{EV}}$ + $\mathsf{sk}_{\mathsf{Sh}}$	$\mathsf{pk}_{\mathrm{EV}}$ + $\mathsf{pk}_{\mathsf{Sh}}$	$\mathsf{sk}_{\mathrm{EV}}$	$\mathsf{pk}_{\mathrm{EV}}$	$\sigma_2^{\mathrm{EV}} + \sigma^{\mathsf{Sc}}$	π_2^{EV}
EvER and Schnorr	4	2	2	1	1	12	5

Fig. 5. Complexity analysis of GUSS (Scheme 4).

the signer and the sanitize sk, pk, ssk and spk, the size of a signature σ and the size of a proof π (second table). We ommit the complexity of the algorithms SaProof and SaJudge since these algorithms are similar to SiProof and SiJudge. The first line corresponds to the generic case, where the values depend on the chosen signature scheme and the chosen verifiable ring signature scheme. The second line corresponds to the case where GUSS is instantiated with Schnorr and EVeR.