Sanitizable Signatures with Several Signers and Sanitizers

Sébastien Canard[1], Amandine Jambert[2], and Roch Lescuyer[1,3]

[1] Orange Labs, Applied Crypto Group, Caen, France
[2] CNIL, Paris, France
[3] ENS, Paris, France

Abstract. Sanitizable signatures allow a signer of a message to give one specific receiver, called a sanitizer, the power to modify some designated parts of the signed message. Most of the existing constructions consider one single signer giving such a possibility to one single sanitizer. In this paper, we formalize the concept with n signers and m sanitizers, taking into account recent models (for 1 signer and 1 sanitizer) on the subject. We next give a generic construction based on the use of both group signatures and a new cryptographic building block, called a trapdoor or proof, that may be of independent interest.

Keywords: Sanitizable signatures, anonymity, trapdoor or proof.

1 Introduction

Cryptographic research provides today a large choice of tools to secure our networks and services. Besides authentication and encryption, it exists several ways to lighten or slightly modify the main cryptographic tools. Regarding signature schemes, it is for example possible to blind the identity of the signer (using e.g. group signatures) or to add properties on the resulting message-signature pair.

Among those variants, the idea of a signature on a document which can be further modified by a designated "sanitizer", without interaction with the signer, has been introduced in [19]. The current definition, introduced in [1] under the named of sanitizable signatures, allows the signer to control which parts of the message can be modified by the chosen sanitizer. The security properties sketched in [1] have been formalized in [7] for the case of one single signer giving the modification power to one single sanitizer[1]: such a scheme should be *transparent* (only the signer and the sanitizer are able to distinguish an original signature from a sanitized one), *immutable* (the sanitizer is unable to modify non admissible blocks of a signed message), *signer-accountable* (a signer can not force a judge to accuse a sanitizer) and *sanitizer-accountable* (a sanitizer can not force a judge to accuse a signer). The notion of *unlinkability* (infeasibility to identify message-signature pairs from the same source) has later been proposed in [8]. Some extensions [18,9] have also been described, allowing the signer to better control the modifications the sanitizer can do.

[1] Even if several sanitizers could exist in the system.

A. Mitrokotsa and S. Vaudenay (Eds.): AFRICACRYPT 2012, LNCS 7374, pp. 35–52, 2012.

It currently exists several sanitizable signature constructions in the literature [1,10,7,9,8,17] which consider *one single* signer allowing *one single* sanitizer to sanitize a given message-signature pair. But nobody has really taken into account the case of multiple signers and sanitizers in a unique system. The closest solutions are either trapdoor solutions [10,21] which allows signer to chose afterwards one sanitizer in a group or the recent work in [8] about unlinkable schemes which can be extended to the case of "one signer and m sanitizers".

Regarding concrete applications, sanitizable signatures with one signer and one sanitizer may be useful in the context of Digital Right Management [10] (signer of a license vs. modifier of a given license), database applications [1] (commercial vendor vs. database administrator) or medical ones [1] but the one-one case does not cover all use cases. The secure routing application proposed in [1] should *e.g.* use a group of n entities acting as both signers and sanitizers. Sanitizable signatures can also be used *e.g.* to protect the privacy of customers in a billing system where the service provider does not obtain the identity of the customer and the billing provider does not know the provided service.

In this paper, we propose the first complete model for sanitizable signatures with n signers and m sanitizers. Our model includes the Brzuska *et al.* [7,8] for 1 signer and 1 sanitizer and considers collusion of adversary.

Ideas behind our model. A signer can choose several designated sanitizers for a given message and each of them is able to modify the resulting message/signature pair. We thus redefine accordingly the notions of accountability and immutability and introduce several notions of transparency: the *no-transparency* where anybody can distinguish a signed message from a sanitized one, the *group transparency* where only signers can make such a distinction and the *full transparency* where this can only be done by the true signer, the true sanitizer and a designated authority.

As there are several signers and sanitizers, we study the case where the signer (resp. the sanitizer) is anonymous within the group of signers (resp. sanitizers). Similarly to the transparency, we also introduce the concept of *group anonymity* where a signer (resp. sanitizer) is anonymous for people who are not in the group of signers (resp. sanitizers). To be complete, we also treat the anonymity revocation by some designated authorities and study the notion of traceability and non-frameability from the group signature world [3,4].

Idea of our construction. Our main (n, m) multi-players sanitizable signature construction is based on the work of Brzuska *et al.* [8], using group signatures [3,5,4,13]. More precisely, we base our new solution on a new cryptographic building block that may be of independent interest: *trapdoor or proof*. Zero-knowledge proofs of knowledge allow one prover to prove to one verifier that she knows some secrets verifying some public relations. An *or proof* enables to prove *e.g.* that the known secret is the discrete logarithm of either y in base g *or* of z in base h. We introduce the concept, and give a practical construction, for a

trapdoor or proof where a given authority can reveal which discrete logarithm is known. Independently, this tool can be used to design electronic voting systems.

The paper is organized as follows. Our model for multi-players sanitizable signature is described in Section 2 (for procedures) and in Section 3 (for security). Section 4 is dedicated to useful tools and Section 5 to our new *trapdoor or proof*. Our main multi-players sanitizable signature scheme is described in Section 6.

2 Multi-players Sanitizable Signatures

Our aim is to propose a model where one signer (among n) can choose a set of sanitizers (among m) such that any sanitizer of the chosen subset is able to sanitize the output message-signature pair. Moreover we want to be consistent with the initial model from Brzuska *et al.* where one signer chooses one sanitizer [7]. We thus keep traditional procedures (Sign, Sanitize, Verify, SigOpen and Judge) and security properties: *immutability*, *signer and sanitizer accountabilities* and *transparency* (see below). We also add the *unlinkability* property [8].

Adding anonymity. As we now have a group of signers and sanitizers, we can handle these groups in different ways. One possibility is to publicly know who is the initial signer (resp. sanitizer) of a given message-signature pair (*no-anonymity*). We can also take the example of group signatures by considering that the signer (resp. sanitizer) can be *anonymous*, except for a designated authority (*full anonymity*). In some cases, the other signers (resp. sanitizers) may need to identify the signer (resp. the sanitizer) of a given message, while this is still not feasible for other parties (*group-anonymity*).

We thus add a full opener $\mathcal{O}_{\text{FULL}}$ who is able to determine (during the FullOpen procedure) the real producer of a given message-signature pair. Similarly, the origin opener $\mathcal{O}_{\text{ORI}}$ is able to retrieve (during the FindOri algorithm) the signer who is at the origin of a given message-signature pair.

The notion of anonymity and the possibility to revoke this property necessary lead to the notion of *traceability* (the identity of an anonymous signer or an anonymous sanitizer can always be retrieved if needed) and *non-frameability* (the infeasibility to produce a wrong opening).

The case of the transparency. The notion of *transparency* says [7] that only the signer and the sanitizer are able to distinguish an original signature from a sanitized one. For this purpose, the signer in [7] has access to a SigOpen procedure which permits her to prove that a given message/signature pair is an original or a sanitized one.

As we consider the case of a group, we introduce, as for anonymity, the notion of *group anonymity* where the the SigOpen procedure is extended to any signer in the group (not only the real signer).

We also keep the traditional notion (called *full transparency*) where only the true original signer, the true sanitizer (if relevant), and a new introduced authority called the algorithm opener $\mathcal{O}_{\text{ALG}}$, are able to prove that one message-signature

pair is an original or not. For the latter, we introduce the AlgOpen procedure which can be executed by $\mathcal{O}_{\text{ALG}}$. Note that the true signer is always able to make such distinction but she is not necessarily able to prove it.

Remark 1. We do not need to add an unforgeability property since it is implied by the accountability, traceability and non-frameability. In fact, from Proposition 4.2 of [7], we obtain that accountabilities imply unforgeability and the given proof still work in our case. Moreover (cf. Appendix A of [4]), the group signature's unforgeability follows from traceability plus non-frameability.

General Definition. A multi-players sanitizable signature scheme involves a set of signers, a set of sanitizers, an issuer $\mathcal{I}$ that may be divided into $\mathcal{I}_{\text{SIG}}$ and $\mathcal{I}_{\text{SAN}}$ and an opener $\mathcal{O}$ that may be divided into $\mathcal{O}_{\text{FULL}}$, $\mathcal{O}_{\text{ALG}}$, and $\mathcal{O}_{\text{ORI}}$.

Given a message m of length ℓ and divided into t blocks, ADM is defined by the signer as (i) the length ℓ_i of each block m_i (such that $\ell = \sum_{i=1}^{t} \ell_i$) and (ii) the index of the block which will be modifiable by the sanitizer, i.e. the subset $\mathcal{T}$ of $[1, t]$ such that for all $i \in \mathcal{T}$, m_i is modifiable. By misuse of notation, we say that $i \in \mathsf{ADM}$ if $i \in \mathcal{T}$. If two messages m_0 and m_1 are defined as having the same admissible parts ADM, we note that $\mathsf{ADM}(\mathsf{m}_0, \mathsf{m}_1) = 1$. On input a message m and the variable ADM, the sanitizer define the modifications MOD as the set of all the (i, m'_i) such that she is able to replace the i-th block of m by m'_i. We say that MOD matches ADM if $\forall i \in \mathsf{MOD}$, $i \in \mathsf{ADM}$.

Definition 1 (Multi-players sanitizable signature scheme). *Let λ be a security parameter. A (n, m)-multi-players sanitizable signature scheme Π is composed of the following eleven algorithms.*

- Setup(1^λ) *outputs the public key* gpk *of the system, the secret key* $\mathsf{isk} := (\mathsf{isk}_{\text{SIG}}, \mathsf{isk}_{\text{SAN}})$ *of some issuers and, in some cases, an additional opening secret key denoted by* $\mathsf{osk} := (\mathsf{osk}_{\text{FULL}}, \mathsf{osk}_{\text{ALG}}, \mathsf{osk}_{\text{ORI}})$.
- SigKG$(1^n, 1^\lambda, \mathsf{isk}_{\text{SIG}})$ *and* SanKG$(1^m, 1^\lambda, \mathsf{isk}_{\text{SAN}})$ *take as input the issuer key* $\mathsf{isk}_{\text{SIG}}$ *(resp.* $\mathsf{isk}_{\text{SAN}}$*), the number n (resp. m) of signers (resp. of sanitizers) and λ. They output two vectors of keys* $(\boldsymbol{sk}_{\text{SIG}}, \boldsymbol{pk}_{\text{SIG}})$ *(resp.* $(\boldsymbol{sk}_{\text{SAN}}, \boldsymbol{pk}_{\text{SAN}})$*). From now on, the whole public key* $(\mathsf{gpk}, \boldsymbol{pk}_{\text{SIG}}, \boldsymbol{pk}_{\text{SAN}})$ *is denoted* PK.
- Sign$(\mathsf{m}, \boldsymbol{sk}_{\text{SIG}}[i], \widetilde{\boldsymbol{pk}}_{\text{SAN}}, \mathsf{ADM}, \mathsf{PK})$ *enables the signer i to sign a message* m *for authorized sanitizers* $\widetilde{\boldsymbol{pk}}_{\text{SAN}} \subseteq \boldsymbol{pk}_{\text{SAN}}$ *according to* ADM *as defined above. It outputs a signature σ on* m*. By convention σ contains* ADM *and* $\widetilde{\boldsymbol{pk}}_{\text{SAN}}$*. Note that σ also contains the way for authorized sanitizers to sanitize* m.
- Sanitize$(\mathsf{m}, \sigma, \boldsymbol{sk}_{\text{SAN}}[j], \mathsf{MOD}, \mathsf{PK})$ *is carried out by the sanitizer j to sanitize a message-signature pair* (m, σ)*. The modifications* MOD *describe the new message* m' *as defined above. This algorithm outputs a new signature σ' and the modified message* m' *or* $\perp$ *in case of error (for example, j is not able to sanitize this message).*
- Verify$(\mathsf{m}, \sigma, \mathsf{PK})$ *allows to verify the signature σ (sanitized or not) on the message* m*. It outputs* 1 *if the signature is correct and* 0 *if it is not.*

FULLOPEN$(\mathsf{m}, \sigma, \mathsf{osk}_{\text{FULL}}, \mathsf{PK})$ *enables the opener* $\mathcal{O}_{\text{FULL}}$ *to find the identity of the producer of the given message. It outputs the string* FULL*, an identity* I_{FULL} *which is either* $(\mathsf{sig}, i_{\text{FULL}})$ *or* $(\mathsf{san}, j_{\text{FULL}})$*, and a proof* τ_{FULL} *of this claim. In case* $I_{\text{FULL}} = 0$*, it is claiming that no one produced* σ*.*

ALGOPEN$(\mathsf{m}, \sigma, \mathsf{osk}_{\text{ALG}}, \mathsf{PK})$ *enables the opener* $\mathcal{O}_{\text{ALG}}$ *to find whether the couple* (m, σ) *is an original or a sanitized couple. It outputs the string* ALG*, next either* $I_{\text{ALG}} = \mathsf{sig}$ *(original signature) or* $I_{\text{ALG}} = \mathsf{san}$ *(sanitized signature), and a proof* τ_{ALG} *of this claim. In case* $I_{\text{ALG}} = 0$*, the result is that the opener* $\mathcal{O}_{\text{ALG}}$ *cannot conclude.*

FINDORI$(\mathsf{m}, \sigma, \mathsf{osk}_{\text{ORI}}, \mathsf{PK})$ *enables the opener* $\mathcal{O}_{\text{ORI}}$ *to find the original signer of the given message. It outputs the string* ORI*, the identity* $I_{\text{ORI}} = (\mathsf{sig}, i_{\text{ORI}})$ *of the original signer and a proof* τ_{ORI} *of this claim. In case* $I_{\text{ORI}} = 0$*, it is claiming that no signer is at the origin of* σ*. Note that* i_{ORI} *is not necessarily the identity of the actor having produced the signature* σ*, since this one may have been sanitized after the original signature from* i_{ORI}*.*

SIGOPEN$(\mathsf{m}, \sigma, (\text{ORI}, I_{\text{ORI}}, \tau_{\text{ORI}}), \boldsymbol{sk}_{\text{SIG}}[\tilde{i}], \mathsf{PK}, \mathsf{DB})$ *enables the signer* $\tilde{i}$ *to be convinced, using an entry* $(\text{ORI}, I_{\text{ORI}}, \tau_{\text{ORI}})$ *(with* $I_{\text{ORI}} := (\mathsf{sig}, i_{\text{ORI}})$*) which could have been produced by the* FINDORI *algorithm, that the signer* i_{ORI} *is the originator of the given message. The signer* $\tilde{i}$ *may use a set* DB *of couples* (m_k, σ_k) *and proves that the given message-signature pair* (m, σ) *is or is not a sanitized pair. It outputs a triple containing the string* SIG*, either* $I_{\text{SIG}} = I_{\text{ORI}}$ *if* (m, σ) *a true signature or* $(\mathsf{san}, 0)$ *if* (m, σ) *was sanitized, and a proof* τ_{SIG} *(including* τ_{ORI}*). It outputs* $I_{\text{SIG}} = 0$ *if the signer* $\tilde{i}$ *can not conclude.*

JUDGE$(\mathsf{m}, \sigma, \mathsf{gpk}, (s, I_s, \tau_s), \mathsf{PK})$ *is a public algorithm which aims at deciding the origin of a given message-signature pair* (m, σ)*. According to the string* $s \in \{\text{FULL}, \text{ALG}, \text{ORI}, \text{SIG}\}$*, it outputs* 1 *if the predicate guessed in* τ_s *is exact and* 0 *otherwise.*

The correctness property states that all of them should be correct, from the verification to the different opening algorithms.

3 Security Requirements

We now give the security definitions a multi-players sanitizable signature scheme should satisfy. Our work is based on those from [7] and [3].

ORACLES. The security properties will be displayed using experiments in which the adversary's attacks are modelled by having access to some oracles. In the following, $\mathcal{CU}$ denotes the set of corrupted users (as a signer or a sanitizer).

- $\mathsf{setup}(\cdot, \cdot, \cdot)$: this oracle corresponds to the generation of the different keys and parameters. It takes as input the parameters $\lambda, n, m \in \mathbb{N}$ and executes the procedures SETUP$(\cdot)$, SIGKG$(\cdot, \cdot, \cdot)$ and SANKG$(\cdot, \cdot, \cdot)$ and the set PK is given on output, while $\mathsf{SK} = \{\mathsf{isk}, \mathsf{osk}, \mathbf{sk}_{\text{SIG}}, \mathbf{sk}_{\text{SAN}}\}$ is kept secret (for now).

– $\mathsf{corrupt}(\cdot,\cdot,\cdot)$: the adversary can corrupts a signer or a sanitizer. This oracle takes as input three elements: the first one $a \in \{\mathsf{sig}, \mathsf{san}\}$ says whether the corrupted player is a signer or a sanitizer, the second argument $k \in \mathbb{N}$ gives the identity of the corresponding signer ($k \in [1, n]$) or sanitizer ($k \in [1, m]$) and the third one corresponds to a public key pk. The couple (a, k) is added to the set $\mathcal{CU}$ and the oracle sets $(\mathbf{pk}_{\text{SIG}}[k], \mathbf{sk}_{\text{SIG}}[k]) = (\mathsf{pk}, \perp)$ if $a = \mathsf{sig}$ (or $(\mathbf{pk}_{\text{SAN}}[k], \mathbf{sk}_{\text{SAN}}[k]) = (\mathsf{pk}, \perp)$ if $a = \mathsf{san}$). An adversary having access to no corruption oracle is denoted $\mathcal{A}^{(0)}$, an adversary only having access to $\mathsf{corrupt}(\mathsf{sig}, \cdot, \cdot)$ (resp. $\mathsf{corrupt}(\mathsf{san}, \cdot, \cdot)$) is denoted $\mathcal{A}^{(\mathsf{si})}$ (resp. $\mathcal{A}^{(\mathsf{sa})}$), while an adversary having access to both is denoted by $\mathcal{A}^{(*)}$.
– $\mathsf{sign}(\cdot,\cdot,\cdot)$, $\mathsf{sanitize}(\cdot,\cdot,\cdot,\cdot)$, $\mathsf{fullopen}(\cdot,\cdot)$, $\mathsf{algopen}(\cdot,\cdot)$, $\mathsf{findorigin}(\cdot,\cdot)$ and finally $\mathsf{sigopen}(\cdot,\cdot,\cdot,\cdot)$: these oracles are related to the procedures given in Definition 1 (without the non necessary public parameters). The set of queries and answers to and from the sign (resp. the $\mathsf{sanitize}$) oracle is denoted $\mathbf{\Sigma}_{\mathsf{sig}}$ (resp. $\mathbf{\Sigma}_{\mathsf{san}}$) and is composed of elements of the form $(\mathsf{m}_k, i_k, \widetilde{\mathbf{pk}}_{\text{SAN},k}, \mathsf{ADM}_k, \sigma_k)$ (resp. $(\mathsf{m}_k, \sigma_k, j_k, \mathsf{MOD}_k, m'_k, \sigma'_k)$).

Adversaries. For each property, there are two types of adversary.

1. A generator adversary $\mathcal{A}_{\mathsf{gen}}$ outputs something that will pass some given criteria. The experiment outputs 1 if all criteria on the adversary's output are verified. For any adversary $\mathcal{A}_{\mathsf{gen}}$ against a property prop and any parameters $\lambda, n, m \in \mathbb{N}$, the success probability of $\mathcal{A}_{\mathsf{gen}}$ is the probability that the experiment outputs 1. We say that the scheme verifies prop if this success is negligible (as a function of λ, n, m) for any polynomial-time $\mathcal{A}_{\mathsf{gen}}$.
2. A choose-then-guess adversary $\mathcal{A} = (\mathcal{A}_{\mathsf{ch}}, \mathcal{A}_{\mathsf{gu}})$ is divided into two phases: $\mathcal{A}_{\mathsf{ch}}$ for the "choose" phase or $\mathcal{A}_{\mathsf{gu}}$ for the "guess" one. For the experiments, a challenge bit $b \in \{0, 1\}$ is set and for any adversary $\mathcal{A}$ against a property prop and any parameters $\lambda, n, m \in \mathbb{N}$, the advantage of $\mathcal{A}$ is $\Pr\left[\mathbf{Exp}^{\mathsf{prop\text{-}1}}_{\Pi,\mathcal{A}} = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{prop\text{-}0}}_{\Pi,\mathcal{A}} = 1\right]$. We next say that the whole scheme verifies the property prop if this advantage is negligible for any polynomial-time $\mathcal{A}$.

Immutability. The immutability says that it is not feasible, for an adversary controlling all the sanitizers, to make a modification on a signed message by a non-authorized sanitizer, to modify a signed message in a non admissible part, or to modify ADM (see [17]). We allow the adversary to corrupt signers, but the output pair should not originally come from a corrupted signer.

$\mathbf{Exp}^{\mathsf{imm}}_{\Pi,\mathcal{A}}(\lambda, n, m)$:
– $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{setup}(1^\lambda)$;
– $(\mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{(*)}_{\mathsf{gen}}(\mathsf{PK}, \mathsf{isk}_{\text{SAN}})$; // *let* $(\widetilde{\boldsymbol{pk}}^*_{\text{SAN}}, \mathsf{ADM}^*) \in \sigma^*$
– $(\text{ORI}, I_{\text{ORI}}, \tau_{\text{ORI}}) \leftarrow \text{FindOri}(\mathsf{m}^*, \sigma^*, \mathsf{osk}_{\text{ORI}}, \mathsf{PK})$
– if $\left[\text{Verify}(\mathsf{m}^*, \sigma^*, \mathrm{PK}) = 0\right]$ or $\left[I_{\text{ORI}} = (\mathsf{sig}, i_{\text{ORI}}) \in \mathcal{CU}\right]$, return 0;
– if $\forall(\mathsf{m}_k, i_{\text{ORI}}, \widetilde{\mathbf{pk}}_{\text{SAN},k}, \mathsf{ADM}_k, \cdot) \in \mathbf{\Sigma}_{\mathsf{sig}}$, $\left[\widetilde{\mathbf{pk}}_{\text{SAN},k} \neq \widetilde{\mathbf{pk}}^*_{\text{SAN}}\right]$ or $\left[\mathsf{ADM}_k \neq \mathsf{ADM}^*\right]$ or $\left[\exists \ell \in [1, t_k] \text{ s.t. } \mathsf{m}_k[\ell] \neq m^*[\ell] \text{ and } \ell \notin \mathsf{ADM}_k\right]$, then return 1.

SANITIZER ACCOUNTABILITY. The adversary controls all the sanitizers and outputs a (m^*, σ^*) pair which will be attributed to a signer, while this is not the case. The first possibility for the adversary is to output a valid tuple (ALG, sig, τ_{ALG}) accepted by the judge. The second possibility is to make use of an honest signer i^* of its choice, such that when i^* executes the SIGOPEN algorithm, the output is (SIG, I_{sig}) with I_{sig} being an honest signer. Since the adversary is given the ability to corrupt signers, we should be convinced that σ^* neither comes from a corrupted signer nor from the sign oracle.

$\mathbf{Exp}^{\mathsf{san\text{-}acc}}_{\Pi,\mathcal{A}}(\lambda, n, m)$:
– $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{setup}(1^\lambda, n, m)$;
– $(\mathsf{m}^*, \sigma^*, (I^*_{\text{ALG}}, \tau^*_{\text{ALG}}), I^* := (\mathsf{sig}, i^*)) \leftarrow \mathcal{A}^{(\mathsf{*})}_{\mathsf{gen}}(\mathsf{PK}, \mathsf{osk}, \mathsf{isk}_{\text{SAN}})$; // *let* $\widetilde{\boldsymbol{pk}}^*_{\text{SAN}} \in \sigma^*$
– $(\text{ORI}, I_{\text{ORI}}, \tau_{\text{ORI}}) \leftarrow \text{FINDORI}(\mathsf{m}^*, \sigma^*, \mathsf{osk}_{\text{ORI}}, \mathsf{PK})$;
– if $\big[\text{VERIFY}(\mathsf{m}^*, \sigma^*, \mathrm{PK}) = 0\big]$ or $\big[I^* \in \mathcal{CU}\big]$ or $\big[I_{\text{ORI}} = (\mathsf{sig}, i_{\text{ORI}}) \in \mathcal{CU}\big]$ or $\big[\exists(m_k, i_k, \widetilde{\mathbf{pk}}_{\text{SAN},k}, \cdot, \sigma_k) \in \mathbf{\Sigma}_{\mathsf{sig}}$ s.t. $(i_k, \mathsf{m}_k, \widetilde{\mathbf{pk}}_{\text{SAN},k}) = (i_{\text{ORI}}, \mathsf{m}^*, \widetilde{\mathbf{pk}}^*_{\text{SAN}})\big]$, return 0;
– $(\text{SIG}, I_{\text{SIG}}, \tau_{\text{SIG}}) \leftarrow \text{SIGOPEN}(\mathsf{m}^*, \sigma^*, (\text{ORI}, I_{\text{ORI}}, \tau_{\text{ORI}}), \mathbf{sk}_{\text{SIG}}[i^*], \mathsf{PK}, \mathsf{DB})$; where $\mathsf{DB} := \{(m_k, \sigma_k) \mid \forall(\mathsf{m}_k, i_k, \cdot, \cdot, \sigma_k) \in \mathbf{\Sigma}_{\mathsf{sig}}$ s.t. $i_k = i_{\text{ORI}}\}$
– if $\big[\big(I^*_{\text{ALG}} = \mathsf{sig} \wedge \text{JUDGE}(\mathsf{m}^*, \sigma^*, (\text{ALG}, I^*_{\text{ALG}}, \tau^*_{\text{ALG}}, \mathsf{PK}) = 1\big)$ or $\big(I^* \notin \mathcal{CU} \wedge I_{\text{ORI}} = I_{\text{SIG}} \wedge \text{JUDGE}(\mathsf{m}^*, \sigma^*, (\text{SIG}, I_{\text{SIG}}, \tau_{\text{SIG}}), \mathsf{PK}) = 1\big)\big]$, then return 1.

SIGNER ACCOUNTABILITY. The adversary controls all the signers and outputs a (m^*, σ^*) pair which will be attributed to a sanitizer, while this is not the case. The first possibility is to output a judge-accepted tuple (ALG, san, τ_{ALG}). The second possibility is to produce a judge-accepted proof (SIG, $(\mathsf{san}, 0)$, τ_{SIG}), which may be output by the SIGOPEN procedure. Again, we should be convinced that σ^* neither comes from a corrupted sanitizer/signer nor from the $\mathsf{sanitize}$ oracle.

$\mathbf{Exp}^{\mathsf{sig\text{-}acc}}_{\Pi,\mathcal{A}}(\lambda, n, m)$:
– $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{setup}(1^\lambda, n, m)$;
– $(\mathsf{m}^*, \sigma^*, (I^*BOn, \tau^*)) \leftarrow \mathcal{A}^{(\mathsf{*})}_{\mathsf{gen}}(\mathsf{PK}, \mathsf{osk}, \mathsf{isk}_{\text{SIG}})$;
– $(\text{FULL}, I_{\text{FULL}}, \tau_{\text{FULL}}) \leftarrow \text{FULLOPEN}(\mathsf{m}^*, \sigma^*, \mathsf{osk}_{\text{FULL}}, \mathsf{PK})$;
– if $\big[\text{VERIFY}(\mathsf{m}^*, \sigma^*, \mathsf{PK}) = 0\big]$ or $\big[I_{\text{FULL}} = (\mathsf{san}, j^*) \in \mathcal{CU}\big]$ or $\big[\exists(\cdot, \sigma_k, \cdot, \cdot, m'_k, \cdot) \in \mathbf{\Sigma}_{\mathsf{san}}$ s.t. $(\mathbf{pk}_{\text{SAN}}[j^*] \in \widetilde{\mathbf{pk}}_{\text{SAN},k}$ or $\mathsf{m}'_k = \mathsf{m}^*)\big]$, return 0;
– if $\big[\big(I^* = \mathsf{san} \wedge \text{JUDGE}(\mathsf{m}^*, \sigma^*, (\text{ALG}, I^*, \tau^*), \mathsf{PK}) = 1\big)$ or $\big([I^* = (\mathsf{san}, 0) \wedge \text{JUDGE}(\mathsf{m}^*, \sigma^*, (\text{SIG}, I^*, \tau^*), \mathsf{PK}) = 1\big)\big]$, then return 1.

TRANSPARENCY. The aim of the adversary is here to decide whether a given message-signature is a sanitized one or not. In the *full transparency* case, she has access to the signer corruption oracle, while she does not in the *group transparency* case. The existence of SIGOPEN obviously implies that the full

transparency can not be reached. Therefore, to design a fully transparent multi-players sanitizable signature scheme, the SIGOPEN procedure must be restricted to the case $\tilde{i} = i_{\text{ORI}}$.

$\mathbf{Exp}^{\mathsf{tran}\text{-}b}_{\Pi,\mathcal{A}}(\lambda, n, m)$ // $b \in \{0,1\}$; $\mathcal{A} = \mathcal{A}^{(*)}$ *if full and* $\mathcal{A} = \mathcal{A}^{(sa)}$ *if group*:
- $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{setup}(1^\lambda, n, m)$;
- $(\mathsf{m}^*, \mathsf{ADM}^*, \mathsf{MOD}^*, i^*, j^*, \widetilde{\mathbf{pk}}^*_{\text{SAN}}, st) \leftarrow \mathcal{A}^{(*)}_{\mathsf{ch}}(\mathsf{PK})$;
- if $i^* \in \mathcal{CU}$ or $j^* \in \mathcal{CU}$, return $\perp$;
- $\sigma^* \leftarrow \text{SIGN}(\mathsf{m}^*, \mathbf{sk}_{\text{SIG}}[i^*], \widetilde{\mathbf{pk}}^*_{\text{SAN}}, \mathsf{ADM}^*, \mathsf{PK})$;
- $(\mathsf{m}'^*, \sigma'^*_0) \leftarrow \text{SANITIZE}(\mathsf{m}^*, \sigma^*, \mathbf{sk}_{\text{SAN}}[j^*], \mathsf{MOD}^*, \mathsf{PK})$;
- if $b = 1$, then $\sigma'^*_1 \leftarrow \text{SIGN}(\mathsf{m}'^*, \mathbf{sk}_{\text{SIG}}[i^*], \widetilde{\mathbf{pk}}^*_{\text{SAN}}, \mathsf{ADM}^*, \mathsf{PK})$;
- $b^* \leftarrow \mathcal{A}^{(*)}_{\mathsf{gu}}(\mathsf{m}'^*, \sigma'^*_b, st)$
- if $(\mathsf{m}'^*, \sigma'^*_b)$ was queried to $\mathsf{sigopen}$, return $\perp$, else return b^*.

UNLINKABILITY. The aim of the adversary is here to choose two messages that become identical once sanitized and decide which one has been sanitized. The adversary has access to a left-or-right oracle which executes the sanitization according to a random bit the adversary must guess.

$\mathbf{Exp}^{\mathsf{unlink}\text{-}b}_{\Pi,\mathcal{A}}(\lambda, n, m)$ // $b \in \{0,1\}$:
- $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{setup}(1^\lambda, n, m)$;
- $(\mathsf{m}^*_0, \mathsf{m}^*_1, \sigma^*_0, \sigma^*_1, \mathsf{MOD}^*_0, \mathsf{MOD}^*_1, j^*_0, j^*_1, st) \leftarrow \mathcal{A}^{(*)}_{\mathsf{ch}}(\mathsf{PK})$;
- $(\mathsf{m}'^*, \sigma'^*) \leftarrow \text{SANITIZE}(\mathsf{m}^*_b, \sigma^*_b, \mathbf{sk}_{\text{SAN}}[j^*_b], \mathsf{MOD}^*_b, \mathsf{PK})$;
- $(\text{ORI}, I_{\text{ORI}}, \tau_{\text{ORI}}) \leftarrow \text{FINDORI}(\mathsf{m}'^*, \sigma'^*_b, \mathsf{osk}_{\text{ORI}}, \mathsf{PK})$;
- if $\left[I_{\text{ORI}} = 0\right]$ or $\left[I_{\text{ORI}} = (\mathsf{sig}, i_{\text{ORI}})$ and $i_{\text{ORI}} \in \mathcal{CU}\right]$ or $\left[j_0 \in \mathcal{CU}\right]$ or $\left[j_1 \in \mathcal{CU}\right]$ or $\left[\text{JUDGE}(m'^*, \sigma'^*_b, (\text{ORI}, I_{\text{ORI}}, \tau_{\text{ORI}}), \mathsf{PK}) = 0\right]$, return $\perp$;
- $b^* \leftarrow \mathcal{A}^{(*)}_{\mathsf{gu}}(\mathsf{m}'^*, \sigma'^*_b, st)$
- if $(\mathsf{m}'^*, \sigma'^*)$ was queried to $\mathsf{sigopen}$, return $\perp$, else return b^*.

TRACEABILITY. The traceability says that the opening should always conclude. The adversary wins if she is able to output a message-signature pair such that the opener ($\mathcal{O}_{\text{FULL}}$, $\mathcal{O}_{\text{ORI}}$) outputs $\perp$ or is unable to produce a correct proof τ of its claim.

$\mathbf{Exp}^{\mathsf{trac}}_{\Pi,\mathcal{A}}(\lambda, n, m)$:
- $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{setup}(1^\lambda, n, m)$;
- $(\mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{(*)}_{\mathsf{gen}}(\mathsf{PK}, \mathsf{osk})$;
- $(\text{FULL}, I_{\text{FULL}}, \tau_{\text{FULL}}) \leftarrow \text{FULLOPEN}(\mathsf{m}^*, \sigma^*, \mathsf{osk}_{\text{FULL}}, \mathsf{PK})$;
- $(\text{ORI}, I_{\text{ORI}}, \tau_{\text{ORI}}) \leftarrow \text{FINDORI}(\mathsf{m}^*, \sigma^*, \mathsf{osk}_{\text{ORI}}, \mathsf{PK})$;
- if $\text{VERIFY}(\mathsf{m}^*, \sigma^*, \mathsf{PK}) = 0$, return 0;
- if $\left[\text{JUDGE}(\mathsf{m}^*, \sigma^*, (s, I_s, \tau_s), \mathsf{PK}) = 0\right]$ or $\left[I_s = 0\right]$, with $s \in \{\text{FULL}, \text{ORI}\}$, then return 1.

Sanitizer anonymity. The adversary here controls all the signers, chooses two sanitizers (j_0^*, j_1^*), a pair (m^*, σ^*) and some MOD^* of her choice. Then the j_b^*-th sanitizer sanitizes the signature (for a uniformly chosen bit b) and the adversary aims at guessing b. In the *full anonymity* case, she has access to the sanitizer corruption oracle, while this is not the case in the *group anonymity*. Note that the "no-" and "group-" anonymity can only be defined if the signer is also viewed as a sanitizer (the contrary not being true) because of the transparency property.

$\mathbf{Exp}^{\mathsf{san\text{-}ano}\text{-}b}_{\Pi,\mathcal{A}}(\lambda, n, m)$ // $b \in \{0,1\}$; $\mathcal{A} = \mathcal{A}^{(*)}$ *if full and* $\mathcal{A} = \mathcal{A}^{(\mathsf{sa})}$ *if group*:

- $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{setup}(1^\lambda, n, m)$;
- $(j_0^*, j_1^*, \mathsf{m}^*, \sigma^*, \mathsf{MOD}^*, st^*) \leftarrow \mathcal{A}^{(\mathsf{si})}_{\mathsf{ch}}(\mathsf{PK}, \mathsf{isk}_{\mathrm{SIG}})$;
- $(\mathsf{m}'^*, \sigma'^*) \leftarrow \textsc{Sanitize}(\mathsf{m}^*, \sigma^*, \mathbf{sk}_{\mathrm{SAN}}[j_b^*], \mathsf{MOD}^*, \mathsf{PK})$;
- if $\left[\textsc{Verify}(\mathsf{m}^*, \sigma^*, \mathsf{PK}) = 0\right]$ or $\left[j_0^* \in \mathcal{CU}\right]$ or $\left[j_1^* \in \mathcal{CU}\right]$, return $\perp$;
- $b^* \leftarrow \mathcal{A}^{(\mathsf{si})}_{\mathsf{gu}}(\mathsf{m}'^*, \sigma'^*, st)$];
- if $(\mathsf{m}'^*, \sigma'^*)$ was queried to $\mathsf{fullopen}$, return $\perp$, else return b^*.

Signer anonymity. The adversary now controls all the sanitizers and aims at distinguish between two signers (i_0^*, i_1^*) of her choice, which one has signed a message m^* according to a chosen ADM^*. We next make the same division as for the sanitizer anonymity part, regarding the corruption possibility for the adversary.

$\mathbf{Exp}^{\mathsf{sig\text{-}ano}\text{-}b}_{\Pi,\mathcal{A}}(\lambda, n, m)$ // $b \in \{0,1\}$; $\mathcal{A} = \mathcal{A}^{(*)}$ *if full and* $\mathcal{A} = \mathcal{A}^{(\mathsf{sa})}$ *if group*:

- $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{setup}(1^\lambda, n, m)$;
- $(i_0^*, i_1^*, \mathsf{m}^*, \widetilde{\mathbf{pk}}^*_{\mathrm{SAN}}, \mathsf{ADM}^*, st) \leftarrow \mathcal{A}^{(\mathsf{sa})}_{\mathsf{ch}}(\mathsf{PK}, \mathsf{isk}_{\mathrm{SAN}})$;
- if $\left[i_0^* \in \mathcal{CU}\right]$ or $\left[i_1^* \in \mathcal{CU}\right]$, return $\perp$;
- $\sigma^* \leftarrow \textsc{Sign}(\mathsf{m}^*, \mathbf{sk}_{\mathrm{SIG}}[i_b^*], \widetilde{\mathbf{pk}}^*_{\mathrm{SAN}}, \mathsf{ADM}^*, \mathsf{PK})$;
- $b^* \leftarrow \mathcal{A}^{(\mathsf{sa})}_{\mathsf{gu}}(\mathsf{m}^*, \sigma^*, st)$;
- if (m^*, σ^*) was queried to $\mathsf{fullopen}$, return $\perp$, else return b^*.

Non-Frameability. The non-frameability property argues that it is not possible for an adversary, even being the openers, to falsely accuse an honest user (signer or sanitizer) from having produced a valid signature. This property is different from the accountability ones since it takes into account the case where some corrupted signers (resp. sanitizers) try to accuse an honest signer (resp. sanitizer). Moreover, we study the case of a false accusation during the FullOpen and FindOri procedures. The adversary does not control all users but can corrupt them, as it wants. It finally outputs a valid (m^*, σ^*) pair and a (i^*, τ^*) pair which could have been output by the FullOpen (resp. FindOri) procedure. She wins if the judge outputs that i^* has truly produced σ^*, while this is not the case.

$\mathbf{Exp}^{\mathsf{nf}}_{\Pi,\mathcal{A}}(\lambda, n, m)$:
– $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{setup}(1^\lambda, n, m)$;
– $(\mathsf{m}^*, \sigma^*, i^*, \tau^*) \leftarrow \mathcal{A}^{(*)}_{\mathsf{gen}}(\mathsf{PK}, \mathsf{isk}, \mathsf{osk})$;
– If $\left[\textsc{Verify}(\mathsf{m}^*, \sigma^*, \mathsf{PK}) = 0\right]$ or $\left[I^* \in \mathcal{CU}\right]$ or $\left[(I^* = (\mathsf{sig}, i^*)$ and $\exists(\mathsf{m}_k, i_k, \cdot, \cdot, \cdot) \in \mathbf{\Sigma}_{\mathsf{sig}}$ s.t. $(i_k, \mathsf{m}_k) = (i^*, \mathsf{m}^*))\right]$ or $\left[(I^* = (\mathsf{san}, j^*)$ and $\exists(\cdot, \cdot, j_k, \cdot, \mathsf{m}'_k, \cdot) \in \mathbf{\Sigma}_{\mathsf{san}}$ s.t. $(j_k, \mathsf{m}'_k) = (j^*, \mathsf{m}^*))\right]$, then return 0.
– If $\exists s \in \{\textsc{full}, \textsc{ori}\}$ s.t. $\textsc{Judge}(\mathsf{m}^*, \sigma^*, (s, I^*, \tau^*)) = 1$, then return 1.

Remark 2. Even if relations exist between security properties, no implication remains. This is less obvious in the relationship between non frameability and accountability but (i) an adversary against accountability and using the AlgOpen procedure to win the experiment is unable to win against the non frameability experiment ; (ii) an adversary against the non-frameability is stronger (as he controls all the issuing keys isk) than an adversary against the accountabilities (who only controls $\mathsf{isk}_{\textsc{san}}$ or $\mathsf{isk}_{\textsc{sig}}$).

The suitability with simple sanitizable signature schemes [7] and the way to add extensions [9] are given in the full version of the paper.

4 Primitives

Before giving a construction, let us begin by describing some cryptographic primitives we will use. Let λ be a security parameter.

Digital signature schemes. We will need a standard signature scheme $\mathcal{S} = (\text{KGn}, \textsc{Sign}, \textsc{Verif})$ specified by algorithms for key generation, signing and verifying. It should satisfy the standard notion of unforgeability under chosen message attack [16]. In a nutshell, the adversary is given the public key and can interact with a signing oracle. Finally, the adversary outputs an attempted forgery (m, σ) and wins if σ is valid, and m was never queried to the signing oracle. We denote by $\mathbf{Succ}^{\mathsf{unf}}_{\mathcal{S},\mathcal{A}}(\lambda)$ the success probability of the adversary $\mathcal{A}$ against $\mathcal{S}$.

Pseudo-random functions. Let $\mathcal{PRF} = (\text{FKGn}, \text{PRF})$ be a pseudo-random function, which is defined by the generation algorithm and the pseudo-random function itself. An adversary $\mathcal{A}$ against such scheme is given access to a random function oracle and outputs a value x_0. After that, a bit b is secretly and randomly chosen. If $b = 0$, the adversary receives the output of the $\mathcal{PRF}$ on x_0. If $b = 1$, the adversary receives a random value. The adversary finally outputs a bit b'. The advantage $\mathbf{Adv}^{\mathsf{prf}}_{\mathcal{PRF},\mathcal{A}}(\lambda)$ of $\mathcal{A}$ is the difference between $1/2$ and the probability that $b' = b$.

Group Signatures. In the following, we will need two different types of group signature schemes. First, a BSZ type group signature scheme [4] and second, a similar concept where we do not want an interactive join protocol between the group manager and a group member, but the non-frameability property. This is

an hybrid model between the BMW model [3] for static groups and the BSZ [4] one for dynamic groups. The non-frameability property is needed to ensure accountability, since the signer needs to produce a signature without the presence of the sanitizers [8].

A group signature scheme $\mathcal{GS}$ is composed of an issuer, an opener and members and is given by a tuple (GKGn,UKGn, Join, [NI-Join, GSKGn,] GSign, GVerif, Open, Judge) described as follows. The join protocol is denoted NI-Join in case of a non-interactive procedure and it is next necessary for each user to execute the GSKGn procedure. If Join is interactive, the latter is not necessary.

GKGn is a probabilistic algorithm which on input 1^λ outputs the key pair $(\mathsf{ik}, \mathsf{gpk}_i)$ of the issuer (sub-procedure called IGKGn), the key pair $(\mathsf{ok}, \mathsf{gpk}_o)$ of the opener (sub-procedure called OGKGn) and the group public key $\mathsf{gpk} = (\mathsf{gpk}_i, \mathsf{gpk}_o)$.

UKGn is a probabilistic algorithm executed by each user i and which on input 1^λ outputs her key pair $(\mathbf{upk}[i], \mathbf{usk}[i])$.

Join is an interactive protocol between the issuer taking on input ik and $\mathbf{upk}[i]$ and user i taking on input $\mathbf{usk}[i]$. The issuer makes a new entry $\mathbf{reg}[i]$ in its registration table $\mathbf{reg}$. The new group member i obtains $\mathbf{msk}[i]$.

[NI-Join is an algorithm executed by the issuer taking on input ik and $\mathbf{mpk} \subseteq \mathbf{upk}$. The issuer outputs its registration table $\mathbf{reg}$.

GSKGn is an algorithm executed by a group member i that on input $\mathbf{usk}[i]$ and $\mathbf{reg}[i]$ outputs a private signing key denoted $\mathbf{msk}[i]$.]

GSign is a probabilistic algorithm that takes on input a message m and a private signing key $\mathbf{msk}[i]$ and outputs a group signature σ on m.

GVerif is an algorithm that on input a message m, a group signature σ and gpk outputs 1 if the signature is valid, and 0 otherwise.

Open is an algorithm which on input a message m, a group signature σ and the opener key ok outputs (in a deterministic way) an integer $i \geq 0$ and (in a probabilistic way) a proof τ that i has produced the signature σ on m. If $i = 0$, then no group member produced σ.

Judge is a deterministic algorithm taking on input a message m, a group signature σ, an integer i, the public key $\mathbf{upk}[i]$ of the entity with identity i and a proof-string τ. It outputs 1 if the proof τ is valid and 0 otherwise.

– **Anonymity.** The anonymity property says that the adversary, given signatures produced by a user (among two of his choice) is not able to guess which users provided the signatures. During the related experiment, $\mathcal{A}$ is given access to ik, can corrupt user, obtain their keys, ask for the opening of group signatures and has access to a challenge oracle which takes as input two non-corrupted member i_0 and i_1 and a message m and outputs the group signature of user i_b, for a bit b set by the experiment. Eventually, $\mathcal{A}$ outputs a bit b'. Next, the advantage $\mathbf{Adv}^{\mathsf{ano}}_{\mathcal{GS},\mathcal{A}}(\lambda)$ of $\mathcal{A}$ is the difference between $1/2$ and the probability that $b' = b$.

– **Traceability.** This property says that the adversary is not able to output a valid group signature such that the opening and judge procedures do not occur

properly. $\mathcal{A}$ is given access to ok and outputs a valid (m, σ) which is accepted by the experiment if either the opening procedure outputs $i = 0$ or the JUDGE procedure cannot succeed. The success probability $\mathbf{Succ}^{\mathsf{trac}}_{\mathcal{GS},\mathcal{A}}(\lambda)$ the adversary $\mathcal{A}$ is next the probability that the experiment accepts.

– **Non-frameability.** An adversary $\mathcal{A}$ is not able to falsely accuse an honest user from having produced a valid group signature. $\mathcal{A}$ is given access to $(\mathsf{ik}, \mathsf{ok})$ and outputs a valid $(\mathsf{m}, \sigma, i, \tau)$ which is accepted by the experiment if i is not corrupted (and her keys are unknown) and the judge accepts the proof τ that i has produced σ while this is not the case. The success probability $\mathbf{Succ}^{\mathsf{nf}}_{\mathcal{GS},\mathcal{A}}(\lambda)$ the adversary $\mathcal{A}$ is next the probability that the experiment accepts.

5 A New Tool: Trapdoor "or" Proof

A Zero Knowledge Proof of Knowledge (ZKPK) is an interactive protocol during which a prover proves to a verifier that he knows a set $(\alpha_1, \ldots, \alpha_q)$ of secret values verifying a given relation R without revealing any information about the known secrets. We denote by $\text{POK}(\alpha_1, \ldots, \alpha_q : \mathsf{R}(\alpha_1, \ldots, \alpha_q))$ such proof of knowledge.

INTRODUCTION. Let $\text{REL} = \{(x, w)\}$ be a binary relation. We first consider the protocol, corresponding to a proof of knowledge for REL, which is played by a prover, taking on input x and a witness w, and a verifier taking on input x. In fact, following [12,20], we consider a set $\mathcal{X} = (x_1, \cdots, x_\ell)$ and a proof of knowledge of the "or" statement where both the prover and the verifier take the common input $\mathcal{X}$, while the prover is also given a private input w_i such that $\exists x_i \in \mathcal{X}$ such that$(x_i, w_i) \in \text{REL}$. Additionally to the witness itself, the verifier should not be able to obtain the index i related to x_i.

In our construction, a designated entity should be able to know which index i is really used by the witness of a user to verify REL, while it is still infeasible for every other actors. To the best of our knowledge, this notion of *trapdoor or proof* does not exist in the literature. However, it can be very useful, as we will see later for our main construction of an (n, m)-sanitizable signature scheme, but also *e.g.* for e-voting where the result of the vote (candidate A "or" candidate B) should not be known, except by authorized scrutineers.

DEFINITIONS. In the following, the above or proof is next denoted $\text{TPOK}(w_i : \exists i \in [1, \ell] | (x_i, w_i) \in \text{REL})$ and the whole system, including the key generation TKGN for the trap, and the "opening" procedure TOPEN, is denoted $\mathcal{TOP} = (\text{TKGN}, \text{TPOK}, \text{TOPEN})$.

As usual, such a proof of knowledge should verify the completeness (a valid prover knowing one such w_i is accepted with overwhelming probability), the soundness (a false prover who does not know any such w_i should be rejected with overwhelming probability) and the honest-verifier zero-knowledge properties (the proof does not reveal any information about the witness).

CIPHER COMMUTING RELATIONS. In the following, we will describe a way to generically design a trapdoor or proof for any relation REL. For this purpose, we need to commute the relation and the encryption procedure of a public key en-

cryption scheme and we thus need to restrict the relations where such commuting operation is possible, which gives us the following definition.

Definition 2 (Cipher commuting relation). *Let λ be a security parameter. Let $\mathcal{E} = (\text{EKG}, \text{ENC}, \text{DEC})$ be a secure probabilistic encryption scheme. Let* REL *be a binary relation. We say that* REL *is a* cipher commuting relation *if for all x, w, for all $(\mathsf{epk}, \mathsf{esk}) \longleftarrow \text{EKG}(1^\lambda)$,*

$$(x, w) \in \text{REL} \iff (\text{ENC}(x, \mathsf{epk}), w) \in \text{REL}.$$

OUR GENERIC CONSTRUCTION. Let λ be a security parameter, $\mathcal{E}$ = (EKG, ENC, DEC) be a secure probabilistic encryption scheme and REL be a cipher commuting relation. We want to design the proof $\text{TPOK}(w_i : \exists i \in [1, \ell] | (x_i, w_i) \in \text{REL})$ where the prover knows w_i such that $(x_i, w_i) \in \text{REL}$.

In a nutshell, we encrypt x_i and use a traditional or proof that the encrypted value is one element related to REL, without revealing which one. We next use the cipher commuting property of REL to prove that the knowledge of a witness which verifies REL with the cipher c_i related to x_i.

Let us first consider that the trap has been generated by executing $(\mathsf{epk}, \mathsf{esk}) \leftarrow \text{EKG}(1^\lambda)$. The proof next works as follows.

1. Computes $c_i = \text{ENC}(x_i, \mathsf{epk})$.
2. Generates the standard honest-verifier zero-knowledge proof with both relations:
   (a) $\text{POK}(x_i : \exists i \in [1, \ell] | c_i = \text{ENC}(x_i, \mathsf{epk}))$ and
   (b) $\text{POK}(w_i, x_i : (\text{ENC}(x_i, \mathsf{epk}), w_i) \in \text{REL})$.

As they are connected with an "and", these two proofs of knowledge can be composed together, using standard techniques [11]. The verifier, knowing the relation REL, the ciphertext c_i and epk, can easily verify the two above POK, using standard techniques. Finally, the owner of esk can easily decrypt c_i to retrieve x_i.

A CONCRETE CONSTRUCTION. Let $\mathbb{G}$ be a group of prime order p. Let u, h be random generators of $\mathbb{G}$ and let v and z be two elements of $\mathbb{G}$. We want to design the trapdoor or proof denoted $\text{TPOK}(\alpha : \exists (b, f)\{(u, v), (h, z)\} | f = b^\alpha)$.

Our solution makes use of a homomorphic encryption scheme π = (KEYGEN, ENC, DEC) such that the trapdoor of our construction is the decryption key dk. The encryption public key is $\mathsf{ek} = a$ and the corresponding secret key is $\alpha \in \mathbb{Z}_p^*$ such that $a = d^\alpha$ where $d \in \mathbb{G}$. A prover having access to *e.g.* the discrete logarithm $x \in \mathbb{Z}_p^*$ of v in base u, that is $v = u^x$, can produce a *trapdoor or proof* as follows, with the ElGamal encryption scheme as an concrete instantiation (see [15]).

1. Encrypt v and u as $c_v = (t_1 = va^w, t_2 = d^w)$ and $c_u = (t_3 = ua^r, t_4 = d^r)$ where $w, r \in \mathbb{Z}_p^*$.
2. Produce a (traditional) proof of knowledge on x, r and w such that:
   (a) the pair of encrypted values corresponds to either (v, u) or (z, h), using a set membership proof: $(t_1/v = a^w \wedge t_3/u = a^r)$ *or* $(t_1/z = a^w \wedge t_3/h = a^r)$ (together with the proof that $t_2 = d^w$ and that $t_4 = d^r$.

(b) using the encrypted value (which satisfies the relation $v = u^x$) and the homomorphic property of the encryption scheme, it is done by producing the proof of knowledge of x such that $c_v = c_u^x$. For this purpose, we use that $t_3^x = u^x a^{rx} = va^{rx} = t_1 a^{rx-w}$.

The final *trapdoor or proof* is composed of (t_1, t_2, t_3, t_4) and the following proof of knowledge:

$$V = \textsc{Pok}\Big(w, r, x, \bar{r} : \big((\tfrac{t_1}{v} = a^w \wedge \tfrac{t_3}{u} = a^r) \vee (\tfrac{t_1}{z} = a^w \wedge \tfrac{t_3}{h} = a^r)\big) \\ \wedge t_2 = d^w \wedge t_4 = d^r \wedge t_1 = t_3^x a^{-\bar{r}} a^w \wedge 1 = t_4^x d^{-\bar{r}}\Big)$$

Anyone in possession of α can retrieve the encrypted pair (v, u) and obtain the known discrete logarithm. We here present the more general case where we need to encrypt both u and v. As in [12,20], the (trapdoor) or proof for a representation can also be treated similarly. We do not detailed the case of a representation but we will use it in the following section.

6 Full Transparent and Fully Anonymous Multi-players Sanitizable Signature

We now describe our *fully transparent* and *fully anonymous* sanitizable signature scheme for several signers and sanitizers.

Following the idea from [8], one user is able to sanitize a message/signature pair if she belongs to a group created by the initial signer and related to this message/signature pair. Next, the principle of our signature is to associate (i) a signature of the signer, as member of a group of signers, on the fixed parts of the message with (ii) a group signature on the admissible parts of the message, on behalf of the new group generated by the signer and (iii) a trapdoor or proof of knowledge of either a certified signer key or a certified sanitizer key. The latter is added to prevent everybody to distinguish a signed message from a sanitized one, except by $\mathcal{O}_{alg}$.

Our scheme is composed of openers $\mathcal{O}_{\text{FULL}}$, $\mathcal{O}_{\text{ALG}}$, $\mathcal{O}_{\text{ORI}}$, a group manager $\mathcal{GM}$ for a group signature scheme and a certification authority denoted $\mathcal{CA}$.

Generation Phases. Let μ be a security parameter. We note $\mathcal{GS}_1$ (resp. $\mathcal{GS}_2$) an interactive-join (resp. non-interactive) group signature scheme, $\mathcal{S}$ a standard signature scheme, $\mathcal{TOP}$ a trapdoor or proof system (cf. Section 5) and $\mathcal{PRF}$ a pseudo-random function.

Setup Phase. The certification authority $\mathcal{CA}$ executes twice the key generation $\mathcal{S}.\text{KG}_\text{N}$ for the standard signature scheme $\mathcal{S}$ to obtain two different keys pairs denoted $(\mathsf{cask}_{\mathsf{si}}, \mathsf{capk}_{\mathsf{si}})$ and $(\mathsf{cask}_{\mathsf{sa}}, \mathsf{capk}_{\mathsf{sa}})$. The group manager $\mathcal{GM}$ executes $\mathcal{GS}_1.\text{IGKG}_\text{N}$, which gives isk and gpk_i. The opener $\mathcal{O}_{\text{FULL}}$ executes $\mathcal{GS}_2.\text{OGKG}_\text{N}$, which gives $\mathsf{osk}_{\text{FULL}}$ and $\mathsf{gpk}_{\text{FULL}}$. The opener $\mathcal{O}_{\text{ALG}}$ executes the $\mathcal{TOP}.\text{TKG}_\text{N}$ algorithm, which gives $\mathsf{osk}_{\text{ALG}}$ and $\mathsf{gpk}_{\text{ALG}}$. The opener $\mathcal{O}_{\text{ORI}}$ executes $\mathcal{GS}_1.\text{OGKG}_\text{N}$, which gives $\mathsf{osk}_{\text{ORI}}$ and gpk_o. In the following, we denote $\mathsf{gpk}_{\mathsf{si}} = (\mathsf{gpk}_i, \mathsf{gpk}_o)$.

To sum up, we have $\mathsf{isk}_{\text{SIG}} = (\mathsf{isk}, \mathsf{cask}_{\mathsf{si}})$, $\mathsf{isk}_{\text{SAN}} = (\mathsf{cask}_{\mathsf{sa}})$ and the general public key $\mathsf{gpk} = (\mathsf{capk}_{\mathsf{si}}, \mathsf{capk}_{\mathsf{sa}}\ \mathsf{gpk}_{\mathsf{si}}, \mathsf{gpk}_{\text{FULL}}, \mathsf{gpk}_{\text{ALG}})$.

Signer Key Generation. Each signer i executes the following. She uses the JOIN interactive protocol with $\mathcal{GM}$ to get her private signing key $\mathbf{msk}[i]$. Next, she executes the key generation for the pseudo-random function to obtain $\mathsf{uk}[i] = \mathcal{PRF}.\text{FKGEN}(1^{\mu})$. Then, she uses the user key generation $\mathcal{GS}_2.\text{UKGN}$ for $\mathcal{GS}_2$ to obtain both $\mathbf{upk}_{si}[i]$ and $\mathbf{usk}_{si}[i]$. Finally, she sends $\mathbf{upk}_{si}[i]$ to $\mathcal{CA}$ with a non-interactive proof of knowledge of the related $\mathbf{usk}_{si}[i]$ and $\mathcal{CA}$ generates $\mathbf{uc}_{si}[i] = \mathcal{S}.\text{SIGN}(\mathbf{upk}_{si}[i], \mathsf{cask}_{\mathsf{si}})$.

Sanitizer Key Generation. Each sanitizer j uses $\mathcal{GS}_2.\text{UKGN}$ to get $(\mathbf{usk}_{sa}[j], \mathbf{upk}_{sa}[j])$ and sends $\mathbf{upk}_{sa}[j]$ to $\mathcal{CA}$ with a non-interactive proof of knowledge of the related $\mathbf{usk}_{sa}[j]$. Then, $\mathcal{CA}$ generates $\mathbf{uc}_{sa}[j] = \mathcal{S}.\text{SIGN}(\mathbf{upk}_{sa}[j], \mathsf{cask}_{\mathsf{sa}})$.

SIGNATURE PROCEDURE. During this procedure, the signer first generates the keys of a new group signature scheme (for herself and the chosen sanitizers). She next produces two different group signatures, the first as a member of the group of signers and the second as a member of the new formed group. Let us consider the i-th signer, $i \in [1, n]$ and let m be the message, divided into t parts, she wants to sign. Following ADM given on input, let $\mathsf{m}_{\mathtt{FIX}}$ be the part of m which will not be sanitizable by the sanitizers. The SIGN procedure is described as follows.

Choice of Sanitizers. The signer chooses a subset $J \subseteq [1, m]$ of sanitizers allowed to modify her message with $\widetilde{\mathbf{pk}}_{\text{SAN}}$ the set of their public keys.

Generation of a Group. The signer creates a group for herself and the chosen sanitizers. For this purpose, she uses the group signature scheme $\mathcal{GS}_2$ with a non-interactive join. More precisely, she computes $rd = \mathsf{PRF}(\mathsf{uk}[i], id_{\mathsf{m}})$ where $id_{\mathsf{m}} = \mathsf{m}_{\mathtt{FIX}} \| \mathsf{ADM}$ is the identifier of the initial message.

She next carries out the key generation algorithm IGKGN of the group signature scheme $\mathcal{GS}_2$, using rd as a random (see also [8]). It gives $\mathsf{isk}[i, id_{\mathsf{m}}]$ and $\mathsf{gpk}_i[i, id_{\mathsf{m}}]$. We note $\mathsf{gpk}[i, id_{\mathsf{m}}] = (\mathsf{gpk}_i[i, id_{\mathsf{m}}], \mathsf{gpk}_{\text{FULL}})$. Then, she uses the non-interactive join procedure NI-JOIN of $\mathcal{GS}_2$ to generate the private signing key for each group members, using rd as a random. We denote it $\mathbf{reg}[i, id_{\mathsf{m}}]$ for the signer and $\mathbf{reg}_{sa}[j, id_{\mathsf{m}}]$ for each sanitizer $j \in J$.

She finally obtains her own membership secret key for this group $\mathbf{msk}[i, id_{\mathsf{m}}]$ thanks to the GSKGN procedure on input $\mathbf{upk}_{si}[i]$ and $\mathbf{reg}[i, id_{\mathsf{m}}]$.

Group Signatures Generation. She computes two group signatures, the first as a signer $\sigma_{\mathtt{fix}} = \mathcal{GS}_1.\text{GSIGN}(\mathbf{msk}[i], \mathsf{m}_{\text{SIG}})$ with $\mathsf{m}_{\text{SIG}} = id_{\mathsf{m}} \| \widetilde{\mathbf{pk}}_{\text{SAN}} \| \mathbf{reg}_{sa}$, the second as a member of the new group $\sigma_{\mathtt{full}} = \mathcal{GS}_2.\text{GSIGN}(\mathbf{msk}[i, id_{\mathsf{m}}], \mathsf{m})$.

Proof of Validity. The signer finally proves, thanks to the non-interactive zero-knowledge proofs of knowledge π, that (i) the above is correctly done and (ii) she is either a signer or a sanitizer.

- $\text{POK}(\mathbf{msk}[i, id_{\mathsf{m}}], \mathbf{upk}_{si}[i], \mathbf{reg}[i, id_{\mathsf{m}}]) : \mathbf{msk}[i, id_{\mathsf{m}}] = \mathcal{GS}_2.\text{GSKGN}(\mathbf{upk}_{si}[i], \mathbf{reg}[i, id_{\mathsf{m}}]))$; and

- $\text{POK}(\mathbf{msk}[i, id_{\mathsf{m}}] : \sigma_{\mathtt{full}} = \mathcal{GS}_2.\text{GSIGN}(\mathbf{msk}[i, id_{\mathsf{m}}], \mathsf{m}))$; and
- $\text{TPOK}(\mathbf{upk}_{si}[i],\ \mathbf{uc}_{si}[i] : \exists \mathbf{sk} \in \{\mathsf{cask_{si}},\ \mathsf{cask_{sa}}\} | \mathbf{uc}_{si}[i] = \mathcal{S}.\text{SIGN}(\mathbf{upk}_{si}[i], \mathbf{sk}))$.

As the proofs are connected with an "and" (which is the case for the trapdoor or proof), these relations can be composed together [11].

The resulting signature is $\sigma = (\pi, \sigma_{\mathtt{FIX}}, \sigma_{\mathtt{FULL}}, \mathsf{ADM}, \widetilde{\mathbf{pk}}_{\text{SAN}}, \mathsf{w}_i, \mathbf{reg}_{sa})$ where $\mathbf{reg}_{sa}$ allows sanitizers to obtain their group member keys afterwards.

SANITIZATION PROCEDURE. The sanitization algorithm consists, for the sanitizer, in (i) the creation of a new $\sigma'_{\mathtt{FULL}}$, according to her own keys and the modified message, and (ii) the construction of the corresponding modified proof π'. Let us consider the j-th sanitizer, $j \in [1, m]$, let m be the initial message, with fixed part $\mathsf{m}_{\mathtt{FIX}}$, and let $\sigma = (\pi, \sigma_{\mathtt{FIX}}, \sigma_{\mathtt{FULL}}, \mathsf{ADM}, \widetilde{\mathbf{pk}}_{\text{SAN}}, \mathsf{w}_i, \mathbf{reg}_{sa})$ be a signature on m and MOD be instructions for a new message m'. First of all, if $\mathbf{pk}_{\mathsf{san}}[j] \notin \widetilde{\mathbf{pk}}_{\text{SAN}}$, then the algorithm returns $\perp$. Otherwise, she executes the following steps.

Proof of Group Membership. The sanitizer retrieves its value $\mathbf{reg}_{sa}[j, id_{\mathsf{m}}]$ in $\mathbf{reg}_{sa}$. She executes the GSKGN procedure on input $\mathbf{upk}_{sa}[j]$ and $\mathbf{reg}_{sa}[j, id_{\mathsf{m}}]$ to compute $\mathbf{msk}[j, id_{\mathsf{m}}]$. Then, she produces a new group signature $\sigma'_{\mathtt{full}} = \mathcal{GS}_2.\text{GSIGN}(\mathbf{msk}[j, id_{\mathsf{m}}], \mathsf{m}')$ as a member of the authorized modifier.

Proof of validity. She next produces a proof π' as a sanitizer :

- $\text{POK}(\mathbf{msk}[j, id_{\mathsf{m}}], \mathbf{upk}_{sa}[i], \mathbf{reg}_{sa}[j, id_{\mathsf{m}}]) :$
  $\mathbf{msk}[j, id_{\mathsf{m}}] = \mathcal{GS}_2.\text{GSKGN}(\mathbf{upk}_{sa}[j],\ \mathbf{reg}_{sa}[j, id_{\mathsf{m}}]))$; and
- $\text{POK}(\mathbf{msk}[j, id_{\mathsf{m}}] : \sigma_{\mathtt{full}} = \mathcal{GS}_2.\text{GSIGN}(\mathbf{msk}[j, id_{\mathsf{m}}], \mathsf{m}'))$; and
- $\text{TPOK}(\mathbf{upk}_{sa}[j],\ \mathbf{uc}_{sa}[j] : \exists \mathbf{sk} \in \{\mathsf{cask_{si}},\ \mathsf{cask_{sa}}\} | \mathbf{uc}_{sa}[j] = \mathcal{S}.\text{SIGN}(\mathbf{upk}_{sa}[j], \mathbf{sk}))$.

The resulting signature is $\sigma' = (\pi', \sigma_{\mathtt{FIX}}, \sigma'_{\mathtt{FULL}}, \mathsf{ADM}, \widetilde{\mathbf{pk}}_{\text{SAN}}, \mathsf{w}_i, \mathbf{reg}_{sa})$.

VERIFICATION AND OPENING PROCEDURES. VERIFICATION. On input a signature $\sigma = (\pi, \sigma_{\mathtt{FIX}}, \sigma_{\mathtt{FULL}}, \mathsf{ADM}, \widetilde{\mathbf{pk}}_{\text{SAN}}, \mathsf{w}_i, \mathbf{reg}_{sa})$ on a message m, the verification procedure simply checks both signatures $\sigma_{\mathtt{FIX}}$ and $\sigma_{\mathtt{FULL}}$ and the whole proof π. If all is correct, she outputs 1, otherwise 0.

Opening. We finally describe the different opening procedures for a signature σ as defined above. The ALGOPEN procedure is simply executed by using the trap of the trapdoor or proof as shown in Section 5 with the key $\mathsf{osk}_{\text{ALG}}$. The FINDORI procedure is the execution of the $\mathcal{GS}_1$.OPEN algorithm related to the group signature scheme for signers. Next, the SIGOPEN is executed as described in [8], by using the pseudo-random function and the opening algorithm of the group signature scheme. The FULLOPEN algorithm corresponds to $\mathcal{GS}_2$.OPEN.

SECURITY THEOREM. We finally give the following security theorem.

Theorem 1. *Our full transparent and fully anonymous multi-players sanitizable signature verifies all the required security properties, assuming that the used group signature, the pseudo-random function, the signature scheme (underlying the trapdoor or proof) and the trapdoor or proof are secure.*

Proof (sketch, see full paper for the full proof). Several parts of the proof (immutability, unlinkability and accountabilities) are similar to the the one given in [8], except that we have to replace the unforgeability of the used signature scheme by the traceability and non-frameability of the group signature scheme $\mathcal{GS}_2$. The anonymity properties are given by the anonymity of the group signature scheme, together with the zero-knowledge property of our trapdoor or proof. The non-frameability and traceability properties are related to the ones related to the used group signature schemes, together with the unforgeability of $\mathcal{CA}$'s signature scheme. The full transparency is obtained according to the anonymity of the group signature scheme and the zero-knowledge property of the trapdoor or proof. □

Dealing with the group anonymity. The group anonymity states that the anonymity of sanitizers (resp. signers) is preserved, except for the sanitizers (resp. signers). In this case and regarding the above construction, each sanitizer (resp. signer) should be able to independently decrypt the same message, corresponding to the one related to the full opener or the origin opener: we need a *multi receiver encryption scheme*.

In such a scheme, a designated authority having a master key generates all the "receivers" (sanitizers or signers) secret keys. Such concept already exists, it is named broadcast encryption [14] when it includes a revocation mechanism, traitor tracing [6] when it treats the case of fraudulent receivers or multi-recipient encryption [2] when several messages are encrypted for several recipients. Our need is close to public key traitor tracing scheme, except we do not necessarily need a tracing procedure. Thus a practical construction can be obtained with [6].

Acknowledgments. This work has been supported by the European Commission under Contract ICT-2007-216676 ECRYPT II. We are grateful to the anonymous referees for their valuable comments.

References

1. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable Signatures. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
2. Bellare, M., Boldyreva, A., Staddon, J.: Randomness Re-use in Multi-recipient Encryption Schemeas. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 85–99. Springer, Heidelberg (2002)
3. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)

4. Bellare, M., Shi, H., Zhang, C.: Foundations of Group Signatures: The Case of Dynamic Groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)
5. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
6. Boneh, D., Franklin, M.: An Efficient Public Key Traitor Scheme (Extended Abstract). In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 338–353. Springer, Heidelberg (1999)
7. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of Sanitizable Signatures Revisited. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 317–336. Springer, Heidelberg (2009)
8. Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Unlinkability of Sanitizable Signatures. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 444–461. Springer, Heidelberg (2010)
9. Canard, S., Jambert, A.: On Extended Sanitizable Signature Schemes. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 179–194. Springer, Heidelberg (2010)
10. Canard, S., Laguillaumie, F., Milhau, M.: *Trapdoor* Sanitizable Signatures and Their Application to Content Protection. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 258–276. Springer, Heidelberg (2008)
11. Chaum, D., Pedersen, T.P.: Transferred Cash Grows in Size. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 390–407. Springer, Heidelberg (1993)
12. Cramer, R., Damgrard, I., Schoenmakers, B.: Proof of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
13. Delerablée, C., Pointcheval, D.: Dynamic Fully Anonymous Short Group Signatures. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 193–210. Springer, Heidelberg (2006)
14. Fiat, A., Naor, M.: Broadcast Encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
15. El Gamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
16. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)
17. Gong, J., Qian, H., Zhou, Y.: Fully-Secure and Practical Sanitizable Signatures. In: Lai, X., Yung, M., Lin, D. (eds.) Inscrypt 2010. LNCS, vol. 6584, pp. 300–317. Springer, Heidelberg (2011)
18. Klonowski, M., Lauks, A.: Extended Sanitizable Signatures. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 343–355. Springer, Heidelberg (2006)
19. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS 2000. The Internet Society (2000)
20. De Santis, A., Di Crescenzo, G., Persiano, G., Yung, M.: On monotone formula closure of SZK. In: FOCS 1994, pp. 454–465. IEEE (1994)
21. Yum, D.H., Seo, J.W., Lee, P.J.: Trapdoor Sanitizable Signatures Made Easy. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 53–68. Springer, Heidelberg (2010)