Scope of Security Properties of Sanitizable Signatures Revisited

Hermann de Meer*‡, Henrich C. Pöhls†‡, Joachim Posegga†‡ and Kai Samelin*‡
*Chair of Computer Communication and Computer Networks, University of Passau, Innstr. 43 D-94032 Passau, Germany
Email: demeer@fim.uni-passau.de, ks@sec.uni-passau.de
†Chair of IT-Security, University of Passau, Innstr. 43 D-94032 Passau, Germany
Email: hp@sec.uni-passau.de, jp@sec.uni-passau.de
‡Institute of IT-Security and Security Law (ISL), University of Passau, Innstr. 43 D-94032 Passau, Germany

***Abstract*—Sanitizable signature schemes allow for altering signed data in a signer-controlled way by a semi-trusted third party. This is contrary to standard digital signature schemes, which do not permit any modifications by any party without invalidating the signature. Due to transparency, a strong privacy notion, outsiders cannot see if the signature for a message was created by the signer or by the semi-trusted party. Accountability allows the signer to prove to outsiders if a message was original or touched by the semi-trusted party. Currently, block-level accountability requires to drop transparency. We allow for accountability for sanitizable signatures with transparency on the block-level. Additionally, we generalize the concept of block-level properties to groups. This offers a even more fine-grained control and leads to more efficient schemes. We prove that group-level definitions imply both the block-level and message-level notions. We derive a provably secure construction, achieving our enhanced notions. A further modification of our construction achieves efficient group-level non-interactive public accountability. This construction only requires a constant amount of signature generations to achieve this property. Finally, we have implemented our constructions and the scheme introduced by *Brzuska* et al. at PKC '09 and provide a detailed performance analysis of our reference implementations.**

I. Introduction

Non-malleable signature schemes like RSA-PSS [5], [25] do not allow for any subsequent modification of the protected string of Bits. This behavior ensures that a third party can verify the integrity and authenticity of the signed message. However, there are many scenarios where signed data must be altered by a third party in a *controlled* way. Consider the use of a driver's license as a proof of age. A digital driver's license is digitally signed by the issuing state. To protect the privacy of the license holder, the holder is allowed to anonymize its name, while the date of birth can still be verified by any other party, e.g., a bouncer. Moreover, the government cannot be involved every time a document needs this type of privacy preserving alteration of certain parts. Hence, a third party must be able to alter data without interacting with the original signer. This constellation is known as the "digital document sanitization problem", as formulated by *Miyazaki* et al. [23].

Sanitizable signature schemes (SanSigs), introduced by *Ateniese* et al. [2], explicitly allow for controlled modifications of a signed message. In particular, a SanSig allows that a signed message $m = (m[1], m[2], \cdots, m[\ell])$ can be changed to a different message m'. For each *block* $m[i] \in \{0,1\}^*$, the signer has to decide whether a sanitization by a semi-trusted third party, called the *sanitizer*, is admissible during signature generation. The sanitization neither requires the signer's private key nor requires any protocol interaction with the signer. Hence, the sanitizer is able to derive a new verifying message-signature pair (m', σ') on its own behalf.

A. Motivation.

One may argue that malleable signatures bear an inherent risk. A semi-trusted party is allowed to change signed data and thus the signer gives up control over the statements that are generated in its name. However, delegation of signing rights has proven to be useful for a variety of application scenarios, ranging from sanitizing medical records to secure routing [2], [8], [16]. The security model for accountable sanitizable signatures, introduced in [2], formalized and extended in [7], only allows to decide which party is accountable for the *complete* message-signature pair (m, σ), using additional information provided by the signer. In turn, the notion of *non-interactive public accountability* was introduced in [10]. A non-interactive publicly accountable SanSig allows that *every* third party is able to decide which party is accountable for a given message-signature pair (m, σ), without requiring any additional information besides what is given from the signature. If the accountable party cannot be derived without the auxiliary information, the scheme is said to be *transparent* [2]. In the same paper, *Brzuska* et al. introduced the paradigm of treating properties on the block-level [10]. In particular, they derive the notion of *block-level* non-interactive public accountability, i.e., a third party can decide which party is accountable for *each block* $m[i]$. They require to sacrifice transparency; our construction keeps this stronger privacy notion. Hence, we achieve block-level interactive accountability and transparency.

There are many application scenarios for defining block-level notions. From a legal perspective, it is helpful to know which parts of a given message have been sanitized; even if one part of a message is altered, other parts technically proven to be original may still provide evidence in a legal argument. As an example, consider the case where a department head orders office supplies. The secretary is allowed to adjust the amount of ordered supplies. After the order is placed and a dispute arises over too many ordered items of one kind, it is required to know which party is responsible for the quantity *for each* ordered item. Based on this, the dispute – whether the department head or the secretary is responsible – can be settled. This can be done non-interactively, as the existence of the sanitizer has no

major impact on the privacy concerns of the involved parties. However, sometimes, the knowledge whether a sanitizer has sanitized a message may lead to discrimination, e.g., if a criminal's record is cleared after a few years. In these cases, the existence of the sanitizer must be hidden, i.e., the scheme used must be transparent. However, transparency and non-interactive public accountability are mutually exclusive [10]. Moreover, the current notion of block-level non-interactive public accountability requires the use of linearly many signatures based on the amount of blocks in a message [10]. We therefore generalize the concept of block-level properties to group-level properties, which leads to a reduced complexity in many scenarios. Consider the case of ordering office supplies once more: it may be sufficient to derive the accountability office-wise instead of item-wise. In most scenarios, it still allows for meaningful accountability. Our new generalized definitions contain already existing notions as a border-case. In other words, our work offers generalization and consolidation of the state-of-the-art and allows to use the ideas of [10] but offers the stronger privacy guarantee of transparency. Hence, we unite both approaches.

B. State of the Art.

The standard security properties of SanSigs have first been introduced by *Ateniese* et al. [2]. They have later been formalized and extended by *Brzuska* et al. [7]. Limiting sanitizers to certain values has also been discussed [11], [16], [18], [24]. Later, *Brzuska* et al. introduced the concept of unlinkability, a privacy notion which prohibits a third party from linking two messages [9]. Currently, the notion of unlinkability combined with transparency requires the more costly utilization of group signatures [9]. We thus focus on the security properties presented in [7]. In particular, unlike *Canard* et al. [12], [13], the signer needs to define which blocks are admissible during the signature generation, while we focus on a setting of a single signer and a single sanitizer, as transparent SanSigs for more than one sanitizer currently also require the use of more expensive group signatures [9], [12]. We do note that our ideas remain applicable in unlinkable or multi-sanitizer environments without any adjustments.

In this paper, we focus on SanSigs. Other types of malleable signature schemes exists: redactable signatures, introduced by *Johnson* et al. [17] and in a slightly different way by *Steinfeld* et al. [28]; and the concept of proxy signatures, introduced in [21]. In redactable signature schemes, blocks cannot be modified: they can only be removed, i.e., redacted. Redactable signatures have also been discussed in many varieties, e.g., for quoting [1], [3], [4], arbitrary redactions [14], [22], [26] and more complex data structures like trees [6], [20], [27]. Proxy signatures allow for delegating the signing rights entirely, while sanitizable signatures allow to *alter* a specific message. Due to their different goals we do not discuss proxy or redactable signatures in any more depth.

C. Our Contribution and Outline.

Current block-level accountability notions require at least linearly many signatures, in terms of the number of blocks. We therefore generalize the idea of block-level properties to group-level notions. This allows blocks to be grouped together, which results in a significant performance increase: we only require linearly many operations for the number of groups, not blocks. Hence, we close existing gaps and generalize and merge existing ideas. Moreover, we have implemented our constructions and the SanSig introduced by *Brzuska* et al. [7]. We provide a performance analysis of our implementations for comparison of their performance in this paper. Finally, we give a positive answer to the question posed by *Brzuska* et al. [10]: we prove that it is possible to combine block/group-level accountability and transparency with standard primitives in an efficient way. In particular, we formalize the notion of group-level accountability for transparent sanitizable signatures and give a provably secure construction based on standard signature schemes and tag-based chameleon hashes [7], [19]. An alteration of our construction allows to achieve group-level non-interactive public accountability equal to [10], with only a constant amount of signature generations. This is required, as transparent schemes are not legally recognized by law [10].

II. Preliminaries

We shortly revisit the utilized algorithms, nomenclature and notations. They are derived from [7], but have been extended to allow for group-level notions. We introduce additional algorithms in Sect. III. For a message $m = (m[1], \cdots, m[\ell])$, we call $m[i] \in \{0,1\}^*$ a *block*. "," denotes a uniquely reversible concatenation, while $\perp \notin \{0,1\}^*$ denotes a special symbol not being a string, e.g., to indicate an error or an exception.

ADM describes the sanitizable blocks. W.l.o.g. we assume that ADM contains the total number of blocks in m, denoted by ℓ, and a list of the indices of the modifiable blocks. By including ℓ in ADM, we inhibit all attacks that maliciously try to append or remove blocks at the beginning or end.

GRP contains a set of sets which expresses which admissible blocks $m[i]$ are grouped together. In particular, we have $\text{GRP} \subseteq 2^{\mathbb{N}}$. We require that the elements of GRP are pairwise disjunct, i.e., $\forall i, j, i \neq j : \text{GRP}_i \cap \text{GRP}_j = \emptyset$. Moreover, $|\bigcup_{s_i \in \text{GRP}} s_i| = |\text{ADM}|$ must yield. In other words, every *admissible* block belongs to exactly one group. To clarify this, let $\text{GRP} = \{\{1,5\},\{3,4,6\}\}$. This means, that $\text{GRP}[1] = (m[1], m[5])$ and $\text{GRP}[2] = (m[3], m[4], m[6])$. For simplicity we also use $\text{GRP}[i]$ to denote the uniquely reversible concatenation of each block in $\text{GRP}[i]$. We order the set by order of appearance of the ordered blocks. The cardinality of GRP, i.e., the number of groups, is denoted as γ. Hence, in our example, $\gamma = 2$. To simplify the algorithmic description every non-admissible block belongs to the special group $\text{GRP}[0]$. Hence, in our prior example we have $\text{GRP}[0] = (m[2])$, if $\ell = 6$. Further, we assume that ADM and GRP can *always* be correctly reconstructed from σ, which accounts for the work done in [15].[1]

A secure SanSig consists of the following algorithms:

Definition 1 (Sanitizable Signature Scheme): Any SanSig consists of at least seven PPT algorithms $(\mathsf{KGen}_{\text{sig}}, \mathsf{KGen}_{\text{san}}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$:

1) Key Generation: There are two key generation algorithms, one for the signer and one for the sanitizer. Both create a pair of keys consisting of a private

[1]The notation of GRP can be integrated into ADM. However, for historical reasons, we keep them separate and preserve ADM's original meaning.

key and the corresponding public key, based on the security parameter λ:

$$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$$

$$(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$$

2) Signing: The Sign algorithm takes as input the security parameter λ, a message $m = (m[1], \cdots, m[\ell])$, $m[i] \in \{0,1\}^*$, the secret key sk_{sig} of the signer, the public key pk_{san} of the sanitizer, as well as ADM and GRP. It outputs the message m and a signature σ (or $\perp$, indicating an error):

$$(m, \sigma) \leftarrow \mathsf{Sign}(1^\lambda, m, sk_{\text{sig}}, pk_{\text{san}}, \text{ADM}, \text{GRP})$$

3) Sanitizing: Algorithm Sanit takes the security parameter λ, a message $m = (m[1], \cdots, m[\ell])$, $m[i] \in \{0,1\}^*$, a modification instruction MOD, a signature σ, the public key pk_{sig} of the signer and the secret key sk_{san} of the sanitizer. It modifies the message m according to the modification instruction MOD. We model MOD to contain a list of pairs $(i, m[i]')$, indicating that block i shall be modified into the string $m[i]'$. Note, MOD can be empty or the string $m[i]'$ can be equal to $m[i]$. This allows the sanitizer to take accountability for a given group without modifying it. For simplicity, we write $\text{GRP}[j] \in \text{MOD}$, if at least one block $j \in \text{GRP}[i]$ is to be modified. Sanit generates a new signature σ' for the modified message $m' = \text{MOD}(m)$. Then Sanit outputs m' and σ' (or $\perp$ in case of an error):

$$(m', \sigma') \leftarrow \mathsf{Sanit}(1^\lambda, m, \text{MOD}, \sigma, pk_{\text{sig}}, sk_{\text{san}})$$

4) Verification: The Verify algorithm outputs a decision $d \in \{\texttt{true}, \texttt{false}\}$, indicating the correctness of a signature σ for a message m with respect to the public keys pk_{sig} and pk_{san}.

$$d \leftarrow \mathsf{Verify}(1^\lambda, m, \sigma, pk_{\text{sig}}, pk_{\text{san}})$$

5) Proof: The Proof algorithm takes as input the security parameter λ, the secret signing key sk_{sig}, a message $m = (m[1], \cdots, m[\ell])$, $m[i] \in \{0,1\}^*$ and a signature σ as well a set of (polynomially many) additional message-signature pairs $\{(m_i, \sigma_i) | i \in \mathbb{N}\}$ and the public key pk_{san}. It outputs a string $\pi \in \{0,1\}^*$ (or $\perp$ in case of an error):

$$\pi \leftarrow \mathsf{Proof}(1^\lambda, sk_{\text{sig}}, m, \sigma, \{(m_i, \sigma_i) | i \in \mathbb{N}\}, pk_{\text{san}})$$

6) Judge: Algorithm Judge takes as input a message $m = (m[1], \cdots, m[\ell])$, $m[i] \in \{0,1\}^*$ and a valid signature σ, the public keys of the parties and a proof π. It outputs a decision $d \in \{\texttt{Sig}, \texttt{San}, \perp\}$, indicating whether the message-signature pair has been created by the signer or the sanitizer (or $\perp$ in case of an error):

$$d \leftarrow \mathsf{Judge}(1^\lambda, m, \sigma, pk_{\text{sig}}, pk_{\text{san}}, \pi)$$

We require the usual correctness properties to hold. In particular, every genuinely signed or sanitized message verifies as valid. Moreover, every genuinely created proof makes the judge decide in favor of the signer. See [7] for a formal definition.

For the following definitions, we require that a public key must be efficiently derivable from its corresponding secret key.

Ateniese et al. introduced a set of desirable properties [2], later formalized by *Brzuska* et al. [7], [9], [10]. We list the informal enumerate of all of them for the paper to be self-contained:

- *Unforgeability* assures that third parties cannot produce a signature for a "fresh" message. Fresh means the message has not been signed by the signer, nor issued by the sanitizer. This is similar to the unforgeability requirements of standard signature schemes [7].
- *Immutability* prevents the sanitizer from modifying blocks not admissible [7].
- *Privacy*, prevents third parties from recovering any original information from sanitized message parts. Its extension *unlinkability* [9] describes the "impossibility to use the signatures to identify sanitized message-signature pairs originating from the same source" [9].
- *Transparency* prevents third parties to decide which party is accountable for a given message-signature pair (m, σ). This is important, if the existence of a sanitizer must be hidden, e.g., if sanitization leads to disadvantages of any party involved [7].
- *Accountability* makes the origin (signer or sanitizer) of a signature undeniable. Hence, it allows a judge to settle disputes over the origin of a signature [7]. The judge may request additional information from the signer. *Brzuska* et al. distinguish between Signer- and Sanitizer-Accountability [7].
- *Non-Interactive Public Accountability* allows that a third party can always decide which party is accountable for a given *message*-signature pair (m, σ) [10].
- *Block-level Non-Interactive Public Accountability* allows that a third party can always decide which party is accountable for a *block*-signature pair $(m[i], \sigma)$ [10].

We now give the formal definitions of immutability, privacy, (signer- and sanitizer-) accountability, transparency, and block-level public accountability to increase readability of Sect. III, which introduces new properties and implications. Note, we have already altered the definitions to account for the possibility of grouping blocks.

Definition 2 (Immutability): A sanitizable signature scheme SanSig is *immutable*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Immutability}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ given in Fig. 1 returns 1 is negligible (as a function of λ). To break immutability, the adversary must be able to alter blocks not designated to be sanitized, or to make the signature verify under a new public key [7].

Definition 3 (Privacy): A sanitizable signature scheme SanSig is *private*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Privacy}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ given in Fig. 2 returns 1 is negligibly close to $\frac{1}{2}$ (as a function of λ). Here, the adversary has to decide which message was used to produce the desired outcome [7].

Experiment $\mathsf{Immutability}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$(pk^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\text{sig}}, \cdot, \cdot, \cdot), \mathsf{Proof}(sk_{\text{sig}}, \cdot, \cdot, \cdot, \cdot)}(pk_{\text{san}})$
let (m'_i, σ'_i) for $i = 1, \cdots, q$
denote the answers from Sign
return 1, if:
$\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk_{\text{sig}}, pk^*) = \texttt{true}$, and
$\forall i : pk^* \neq pk_{\text{san},i}$ or
$m^*[j_i] \neq m_i[j_i]$, where $j_i \notin \text{ADM}_i$
//shorter messages are padded with $\perp$

Fig. 1. Immutability

Experiment $\mathsf{Privacy}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$(pk_{\text{san}}, pk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$b \leftarrow \{0, 1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\text{sig}}, \cdot, \cdot, \cdot), \mathsf{Sanit}(\cdot, \cdot, \cdot, \cdot, sk_{\text{san}})}_{\mathsf{Proof}(sk_{\text{sig}}, \cdot, \cdot, \cdot, \cdot), \mathsf{LoRSanit}(\cdot, \cdot, \cdot, \cdot, \cdot, sk_{\text{sig}}, sk_{\text{san}}, b)}(pk_{\text{sig}}, pk_{\text{san}})$
where oracle LoRSanit on input of:
$m_{0,i}, \text{MOD}_{0,i}, m_{1,i}, \text{MOD}_{1,i}, \text{ADM}_i, \text{GRP}_i$
if $\text{MOD}_{0,i} \not\subseteq \text{ADM}_i$, return $\perp$
if $\text{MOD}_{1,i} \not\subseteq \text{ADM}_i$, return $\perp$
if $\text{MOD}_{0,i}(m_{0,i}) \neq \text{MOD}_{1,i}(m_{1,i})$, return $\perp$
let $(m_i, \sigma_i) \leftarrow \mathsf{Sign}(1^\lambda, m_{b,i}, sk_{\text{sig}}, pk_{\text{san}}, \text{ADM}_i, \text{GRP}_i)$
return $(m'_i, \sigma'_i) \leftarrow \mathsf{Sanit}(1^\lambda, m_i, \text{MOD}_{b,i}, \sigma, pk_{\text{sig}}, sk_{\text{san}})$
return 1, if $a = b$

Fig. 2. Privacy

Experiment $\mathsf{Sig-Accountability}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda)$
$(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$b \leftarrow \{0, 1\}$
$(pk^*, \pi^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sanit}(\cdot, \cdot, \cdot, \cdot, sk_{\text{san}})}(pk_{\text{san}})$
let (m'_i, σ'_i) for $i = 1, \cdots, q$
denote the answers from the oracle Sanit
return 1, if:
$\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\text{san}}) = \texttt{true}$, and
$(pk^*, m^*) \neq (pk_{\text{sig},i}, m'_i)$ for all $i = 1, \cdots, q$, and
$\mathsf{Judge}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\text{san}}, \pi^*) = \texttt{San}$

Fig. 3. Signer Accountability

Definition 4 (Signer Accountability): A sanitizable signature scheme SanSig is *signer accountable*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Sig-Accountability}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda)$ given in Fig. 3 returns 1 is negligible (as a function of λ). In this game, the adversary has to generate a proof π^* which makes Judge to decide that the sanitizer is accountable, if it is not [7].

Definition 5 (Sanitizer Accountability): A sanitizable signature scheme SanSig is *sanitizer accountable*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{San-Accountability}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda)$ given in Fig. 4 returns 1 is negligible (as a function of λ). In this game, the adversary has to generate a message-signature (m^*, σ^*) which makes Proof generate a proof π, leading the Judge to decide that the signer is accountable, if it is not [7].

Experiment $\mathsf{San-Accountability}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$b \leftarrow \{0, 1\}$
$(pk^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\text{sig}}, \cdot, \cdot, \cdot), \mathsf{Proof}(sk_{\text{sig}}, \cdot, \cdot, \cdot, \cdot)}(pk_{\text{sig}})$
let $(m_i, \text{ADM}_i, pk_{\text{san},i}, \text{GRP}_i)$ and σ_i for $i = 1, \cdots, q$
denote the queries and answers of oracle Sign
$\pi \leftarrow \mathsf{Proof}(1^\lambda, sk_{\text{sig}}, m^*, \sigma^*, \{(m_i, \sigma_i) | 0 < i \leq q\}, pk^*)$
return 1, if:
$\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk_{\text{sig}}, pk^*) = \texttt{true}$, and
$(pk^*, m^*) \neq (pk_{\text{san},i}, m_i)$ for all $i = 1, \cdots, q$, and
$\mathsf{Judge}(1^\lambda, m^*, \sigma^*, pk_{\text{sig}}, pk^*, \pi) = \texttt{Sig}$

Fig. 4. Sanitizer Accountability

Experiment $\mathsf{Transparency}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$b \leftarrow \{0, 1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\text{sig}}, \cdot, \cdot, \cdot), \mathsf{Sanit}(\cdot, \cdot, \cdot, \cdot, sk_{\text{san}}), \mathsf{Proof}(sk_{\text{sig}}, \cdot, \cdot, \cdot, \cdot)}_{\mathsf{Sanit/Sign}(\cdot, \cdot, \cdot, \cdot, sk_{\text{sig}}, sk_{\text{san}}, b)}(pk_{\text{sig}}, pk_{\text{san}})$
where Sanit/Sign for input $m_i, \text{MOD}_i, \text{ADM}_i, \text{GRP}_i$
$\sigma_i \leftarrow \mathsf{Sign}(1^\lambda, m_i, sk_{\text{sig}}, pk_{\text{san}}, \text{ADM}_i, \text{GRP}_i)$,
$(m'_i, \sigma'_i) \leftarrow \mathsf{Sanit}(1^\lambda, m_i, \text{MOD}_i, \sigma_i, pk_{\text{sig}}, sk_{\text{san}})$
if $b = 1$:
$\sigma'_i \leftarrow \mathsf{Sign}(1^\lambda, m'_i, sk_{\text{sig}}, pk_{\text{san}}, \text{ADM}_i, \text{GRP}_i)$,
finally return (m'_i, σ'_i).
return 1, if $a = b$ and $\mathcal{A}$ has not
queried any (m_i, σ_i) output by Sanit/Sign to Proof.

Fig. 5. Transparency

Definition 6 (Transparency): A sanitizable signature scheme SanSig is *proof-restricted transparent*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Transparency}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda)$ given in Fig. 5 returns 1 is negligibly close to $\frac{1}{2}$ (as a function of λ). The basic idea is that the adversary is not able to decide whether it sees a freshly signed signature or a signature created through Sanitize. Note, we have already altered the definitions of [7], [10] to account for our new group-level definitions.

III. New Scope: Properties for Groups of Blocks

In this section, we introduce the notion of group-level accountability. We show how a signer can use our new definition to simulate existing notions. Hence, we do not restate existing block-level definitions here, as they are border-cases of our new definitions. We first give the definition of group-level non-interactive public accountability which does not offer transparency and then group-level accountability with transparency[2]. We are the first to give a construction which allows for block-by-block (or group-by-group resp.) accountability, while fully achieving transparency.

A. Group-level Non-interactive Public Accountability

To simplify the notion of (block-level) public accountability, *Brzuska* et al. define that the algorithm Judge decides upon

[2]Note, as transparency prohibits a third party from deciding who issued the message-signature pair (m, σ), it directly inhibits the instant non-interactive and public form of accountability [10].

Experiment $\mathsf{Group-Pub-Acc}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$(pk^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,sk_{\text{sig}},\cdot,\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,sk_{\text{san}})}(pk_{\text{san}}, pk_{\text{sig}})$
Let $(m_i, \text{ADM}_i, pk_{\text{san},i}, \text{GRP}_i)$ and (m_i, σ_i) for $i = 1, 2, \cdots, k$ be the queries and answers to and from oracle Sign.
Let $(m_j, \text{MOD}_j, \sigma_j, pk_{\text{sig},j})$ and (m'_j, σ'_j) for $j = 1, 2, \cdots, k'$ be the queries and answers to and from oracle Sanit.
return 1 if
$\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk_{\text{sig}}, pk^*) = \texttt{true}$, and
for m_i with $pk_{\text{san},i} = pk^*$, $\exists q$, such that
$\mathsf{GDetect}(1^\lambda, m^*, \sigma^*, pk_{\text{sig}}, pk^*, q) = \texttt{Sig}$
and $\text{GRP}[q]_i \subseteq \text{MOD}_i$
return 1, if
$\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\text{san}}) = \texttt{true}$, and
for m_j with $pk_{\text{sig},j} = pk^*$, $\exists q$ such that
$\mathsf{GDetect}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\text{san}}, q) = \texttt{San}$
and $\text{GRP}[q]_j \not\subseteq \text{MOD}_j$
return 0

Fig. 6. Group-level Non-Interactive Public Accountability

reception of an empty proof, i.e., $\pi = \perp$ [10]. In this paper, we keep their approach for consistency. Formally, we require the algorithm GDetect. It takes as input the security parameter λ, a message m and a valid signature σ together with the sanitizer's public key pk_{san} and the signer's public key pk_{sig}. Most notably, it also takes as an input a group index i and then returns `San` or `Sig`, indicating which party is accountable for the i^{th} group. This is compareable to *Brzuska* et al.'s definition [10].

GDetect is defined as follows: on input of the security parameter λ, a valid message-signature pair (m, σ), the corresponding public keys pk_{sig} and pk_{san}, and the group index i, GDetect outputs the accountable party for group i (or $\perp$ in case of an error).

$$d \leftarrow \mathsf{GDetect}(1^\lambda, m, \sigma, pk_{\text{sig}}, pk_{\text{san}}, i), d \in \{\texttt{San}, \texttt{Sig}, \perp\}$$

Definition 7 (Non-Interactive Public Accountability): A sanitizable signature scheme SanSig together with an algorithm GDetect is *group-level non-interactive publicly accountable*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Group-Pub-Acc}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ given in Fig. 6 returns 1 is negligible (as a function of λ).

B. Group-level Accountability with Transparency

Next, we define accountability with a detail of group-level, while fully preserving transparency. Let us give an informal definition of group-level accountability first:

> A SanSig offers *group-level accountability*, if for all valid message-signature pairs (m, σ) the algorithm Proof outputs a proof π which allows the algorithm GJudge to decide, if the given *group*-signature pair $(\text{GRP}[i], \sigma)$ originates from the signer or from the sanitizer, even in the presence of malicious signers or sanitizers.

As the algorithm Judge only decides the accountability for the complete message/signature pair, we require an additional

Experiment $\mathsf{Group-Signer-Acc}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$
$(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$b \leftarrow \{0, 1\}$
$(pk^*, \pi^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,sk_{\text{san}})}(pk_{\text{san}})$
Let $(m_j, \text{MOD}_j, \sigma_j, pk_{\text{sig},j})$ and (m'_j, σ'_j) for $j = 1, 2, \cdots, k$ be the queries and answers to and from oracle Sanit.
return 1, if:
$\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\text{san}}) = \texttt{true}$, and
$\forall j : pk_{\text{sig},j} \neq pk^*$, or
$\exists j, q : pk_{\text{sig},j} = pk^*$, s.t.
$\mathsf{GJudge}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\text{san}}, \pi^*, q) = \texttt{San}$, and
$(q, \text{GRP}[q]'_j) \notin \text{MOD}_j$

Fig. 7. group-level Signer Accountability

algorithm able to derive it for each group. The additional algorithm GJudge is defined as follows:

$$d_i \leftarrow \mathsf{GJudge}(1^\lambda, m, \sigma, pk_{\text{sig}}, pk_{\text{san}}, \pi, i)$$

To incorporate the standard accountability notion for the message-level, we define that a sanitizer is accountable for a complete message-signature pair (m, σ), if there exists at least one group i, for which the sanitizer has taken accountability. Vice versa, if there exists no group for which the sanitizer has taken accountability, the signer's accountability follows. This is the expected behavior, as originally defined in [7]. For group-level accountability, we now give new definitions, that include the existing definitions as a border case:

Definition 8 (Group-level Signer Accountability): A sanitizable signature scheme SanSig is *group-level signer accountable*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Group-Signer-Acc}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ given in Fig. 7 returns 1 is negligible (as a function of λ). Basically, to win the game the adversary has to generate a tuple $(pk^*, m^*, \sigma^*, \pi^*)$, which leads GJudge to decide that the sanitizer is accountable for a group $\text{GRP}[q] \in m^*$, while it is not.

Definition 9 (Group-level Sanitizer Accountability): A sanitizable signature scheme SanSig is *group-level sanitizer accountable*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Group-Sanitizer-Acc}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ given in Fig. 8 returns 1 is negligible (as a function of λ). Basically, to win the game the adversary has to generate a tuple (pk^*, m^*, σ^*) for which Proof generates a proof π which leads Judge to decide that the signer is accountable for a group $\text{GRP}[q] \in m^*$, while it is not.

Following the reasoning given in [10], we can express the aforementioned constellation by the following theorems:

Theorem 1: Every SanSig which is group-level signer accountable, is also signer accountable.

Theorem 2: Every SanSig which is group-level sanitizer accountable, is also sanitizer accountable.

Moreover, we can easily emulate block-level properties, if we generate a new group for each block. This proves, that our notions are a generalization of existing notions and are stronger. Moreover, message-level properties, as considered by *Brzuska* et al. [7], can easily be achieved by putting all

Experiment $\mathsf{Group-Sanitizer-Acc}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$b \leftarrow \{0,1\}$
$(pk^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\text{sig}}, \cdot, \cdot), \mathsf{Proof}(sk_{\text{sig}}, \cdot, \cdot, \cdot, \cdot)}(pk_{\text{sig}})$
Let $(m_i, \text{MOD}_i, \sigma_i, pk_{\text{san},i}, \text{GRP}_i)$ and (m_i, σ_i) for $i = 1, 2, \cdots, k$ be the queries and answers to and from the oracle Sign.
$\pi \leftarrow \mathsf{Proof}(1^\lambda, sk_{\text{sig}}, m^*, \sigma^*, \{(m_i, \sigma_i) | 0 < i \leq q\}, pk^*)$
return 1, if:
$\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk_{\text{sig}}, pk^*) = \texttt{true}$, and
$\forall i : pk_{\text{sig},i} \neq pk^*$, or
for all $pk_{\text{san},i} = pk^*$, $\exists q$, s.t.
$\exists q : \mathsf{GJudge}(1^\lambda, m^*, \sigma^*, pk_{\text{sig}}, pk^*, \pi, q) = \texttt{Sig}$ and
$(q, \text{GRP}[q]'_i) \in \text{MOD}_i$

Fig. 8. group-level Sanitizer Accountability

admissible blocks into one single group. Hence, all existing definitions are contained as a border-case of our new ones.

Definition 10 (group-level Accountability): A sanitizable signature scheme SanSig is *group-level accountable*, if it is group-level signer accountable and group-level sanitizer accountable.

IV. Constructions

In this section, we derive two new constructions. The first construction achieves group-level accountability with transparency. The second construction allows a more efficient group-level non-interactive public accountability requiring only a constant number of signatures.

A. Prerequisites

All constructions make use of the tag-based chameleon hash by *Brzuska* et al. [7]. In particular, the chameleon hash must be collision-resistant under random tagging-attacks as assumed and shown in [7].

Definition 11 (Chameleon Hash with Tags): A chameleon hash $\mathcal{CH} := (\mathsf{CHKeyGen}, \mathsf{CHash}, \mathsf{CHAdapt})$ with tags consists of three efficient algorithms:

1) CHKeyGen: The algorithm CHKeyGen takes as input the security parameter 1^λ and outputs the key pair required for the chameleon hash:
$$(sk, pk) \leftarrow \mathsf{CHKeyGen}(1^\lambda)$$
2) CHash: The algorithm CHash takes as input the public key pk, a string m to hash, a tag TAG and a randomness $r \in \{0,1\}^\lambda$. It outputs the digest h:
$$h \leftarrow \mathsf{CHash}(1^\lambda, pk, \mathsf{TAG}, m, r)$$
3) CHAdapt: The algorithm CHAdapt takes as input the private key sk, m, m', TAG, TAG$'$, r. It outputs the new randomness r':
$$r' \leftarrow \mathsf{CHAdapt}(1^\lambda, sk, \mathsf{TAG}, m, r, \mathsf{TAG}', m')$$

As usual, we require all correctness properties to hold. In particular, $\mathsf{CHash}(pk, \mathsf{TAG}, m, r) = \mathsf{CHash}(pk, \mathsf{TAG}', m', r')$ must yield, if r' has been generated genuinely using CHAdapt.

Experiment $\mathsf{Rand-Tag}_{\mathcal{A}}^{\mathcal{CH}}(\lambda)$
$(pk, sk) \leftarrow \mathsf{CHKeyGen}(1^\lambda)$
$(\mathsf{TAG}, m, r, \mathsf{TAG}', m', r') \leftarrow \mathcal{A}^{\mathsf{OAdapt}(sk, \cdot, \cdot, \cdot, \cdot)}(pk)$
where oracle OAdapt for the i^{th} query $(\mathsf{TAG}_i, m_i, r_i, m'_i)$ with $\mathsf{TAG}_i \in \{0,1\}^\lambda$
let $\mathsf{TAG}'_i \leftarrow \{0,1\}^\lambda$ and compute
$r'_i \leftarrow \mathsf{CHAdapt}(sk, \mathsf{TAG}_i, m_i, r_i, \mathsf{TAG}'_i, m'_i)$
return (TAG'_i, r'_i)
return 1, if
$(\mathsf{TAG}, m) \neq (\mathsf{TAG}', m')$ and
let $i = 1, \cdots, q$ denote the i^{th} oracle query
$\mathsf{CHash}(pk, \mathsf{TAG}, m, r) = \mathsf{CHash}(pk, \mathsf{TAG}', m', r')$ and
$\forall i, j : \{(\mathsf{TAG}, m), (\mathsf{TAG}', m')\} \neq \{(\mathsf{TAG}_i, m_i), (\mathsf{TAG}'_i, m'_i)\}$
$\wedge \{(\mathsf{TAG}, m), (\mathsf{TAG}', m')\} \neq \{(\mathsf{TAG}'_i, m'_i), (\mathsf{TAG}'_j, m'_j)\}$

Fig. 9. Collision-Resistance against Random Tagging Attacks [7]

Definition 12 (Collision-Resistance vs. Random-Tag Attacks): A tag-based chameleon hash $\mathcal{CH}$ is said to be collision-resistant under random-tagging attacks, if the probability that the experiment depicted in Fig. 9 returns 1 is negligible (as a function of λ) [7].

A concrete secure instantiation is found in [7]. Note, the distribution of r' is computationally indistinguishable from uniform [7].

B. Group-level Accountable and Transparent SanSig

Next, we introduce a provably secure construction which is transparent, private, immutable, group-level accountable and unforgeable.

Our construction uses the ideas by [7], [15]. In particular, each group is hashed using a tag-based chameleon hash. However, instead of using one tag for the complete message m, we use different tags for each group $\text{GRP}[i]$. We utilize a standard UNF-CMA signature scheme $\mathcal{SS} = (\mathsf{SKeyGen}, \mathsf{SSign}, \mathsf{SVerify})$ to generate the final signature. We also require a pseudorandom function $\mathcal{PRF}$ mapping n-bit input on a n-bit output for n-bit keys and a pseudorandom generator $\mathcal{PRG}$ mapping n-bit inputs to $2n$-bit outputs:

a) KGen_{sig}*:* Generate a key pair of the underlying signature algorithm SKeyGen, i.e., $(pk, sk) \leftarrow \mathsf{SKeyGen}(1^\lambda)$. Pick a key $\kappa \leftarrow \{0,1\}^\lambda$ for the $\mathcal{PRF}$. Output $(pk_{\text{sig}}, sk_{\text{sig}}) = (pk, (sk, \kappa))$.

b) KGen_{san}*:* Generate a key pair of the underlying chameleon hash. Output $(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{CHKeyGen}(1^\lambda)$.

c) Sign*:* On input of $m = (m[1], \cdots, m[\ell])$, $m[i] \in \{0,1\}^*$, pk_{san}, sk_{sig}, ADM, and GRP, Sign draw $\gamma + 1$ nonces $n_i \leftarrow \{0,1\}^\lambda$ and compute: $x_i \leftarrow \mathcal{PRF}(\kappa, n_i)$ and $\mathsf{TAG}_i \leftarrow \mathcal{PRG}(x_i)$ for all $i = 0, \cdots, \gamma$. Draw $\gamma + 1$ additional nonces $r_i \leftarrow \{0,1\}^\lambda$. Let:
$$h[i] \leftarrow \mathsf{CHash}(pk_{\text{san}}, \mathsf{TAG}_i, (i, \text{GRP}[i]), r_i)$$
for all $i = 1, \cdots, \gamma$. Now, let:
$$h[0] \leftarrow \mathsf{CHash}(pk_{\text{san}}, \mathsf{TAG}_0, (\mathsf{TAG}_1, \cdots, \mathsf{TAG}_\gamma, m), r_0)$$

Set

$$\sigma_c \leftarrow \mathsf{SSign}(sk, (h[0], \cdots, h[\gamma], \text{GRP}[0], pk_{\text{san}}, \text{ADM}, \text{GRP}))$$

Output (m, σ), where

$$\sigma = (\sigma_c, (\mathsf{TAG}_i)_{0 \le i \le \gamma}, (n_i)_{0 \le i \le \gamma}, \text{ADM}, \text{GRP}, (r_i)_{0 \le i \le \gamma})$$

d) Verify*:* On input of pk_{sig}, pk_{san}, m and

$$\sigma = (\sigma_c, (\mathsf{TAG}_i)_{0 \le i \le \gamma}, (n_i)_{0 \le i \le \gamma}, \text{ADM}, \text{GRP}, (r_i)_{0 \le i \le \gamma})$$

for each $i \in \text{GRP}$ compute:

$$h[i] \leftarrow \mathsf{CHash}(pk_{\text{san}}, \mathsf{TAG}_i, (i, \text{GRP}[i]), r_i)$$

and

$$h[0] \leftarrow \mathsf{CHash}(pk_{\text{san}}, \mathsf{TAG}_0, (\mathsf{TAG}_1, \cdots, \mathsf{TAG}_\gamma, m), r_0)$$

Output:

$$\mathsf{SVerify}(pk, (h[0], \cdots, h[\gamma], \text{GRP}[0], pk_{\text{san}}, \text{ADM}, \text{GRP}), \sigma_c)$$

e) Sanit*:* On input of pk_{sig}, sk_{san}, m, MOD and σ, first check, if the received message-signature pair is valid using Verify. Check, if $\text{MOD} \subseteq \text{ADM}$. If not, stop outputting $\perp$. For each **group** $\text{GRP}[i] \in \text{MOD}$, draw a nonce $n'_i \leftarrow \{0,1\}^\lambda$ and a new tag $\mathsf{TAG}'_i \leftarrow \{0,1\}^{2\lambda}$. If $\text{GRP}[i] \notin \text{MOD}$, the tags, randoms and nonces are copied from the original signature, i.e., $n'_i = n_i$ and $\mathsf{TAG}'_i = \mathsf{TAG}_i$. If $\text{MOD} \neq \emptyset$, draw an additional nonce $n'_0 \leftarrow \{0,1\}^\lambda$ and an additional tag: $\mathsf{TAG}'_0 \leftarrow \{0,1\}^{2\lambda}$. Compute:

$$r'_i \leftarrow \mathsf{CHAdapt}(sk_{\text{san}}, \mathsf{TAG}_i, (i, \text{GRP}[i]), r_i, \mathsf{TAG}'_i, \text{GRP}[i]')$$

for each $\text{GRP}[i] \in \text{MOD}$ and

$$r'_0 \leftarrow \mathsf{CHAdapt}(sk_{\text{san}}, \mathsf{TAG}_0, (\mathsf{TAG}_1, \cdots, \mathsf{TAG}_\gamma, m), r_0, \mathsf{TAG}'_0, (\mathsf{TAG}'_1, \cdots, \mathsf{TAG}'_\gamma, m'))$$

Output (m', σ'), where $m' \leftarrow \text{MOD}(m)$ and

$$\sigma' = (\sigma_c, (\mathsf{TAG}')_{0 \le i \le \gamma}, (n'_i)_{0 \le i \le \gamma}, \text{ADM}, \text{GRP}, (r'_i)_{0 \le i \le \gamma})$$

f) Proof*:* On input of sk_{sig}, m, $\sigma = (\sigma_c, (\mathsf{TAG}_i)_{0 \le i \le \gamma}, (n_i)_{0 \le i \le \gamma}, \text{ADM}, \text{GRP}, (r_i)_{0 \le i \le \gamma})$, pk_{san} and a sequence of message-signature pairs $\{(m_i, \sigma_i) | i \in \mathbb{N}\}$, search for all groups the matching signatures, s.t.:

$$\mathsf{CHash}(pk_{\text{san}}, \mathsf{TAG}_i, (i, \text{GRP}[i]), r_i) = \mathsf{CHash}(pk_{\text{san}}, \mathsf{TAG}'_i, (i, \text{GRP}'[i]), r'_i)$$

Do the same for the outer chameleon hash:

$$\mathsf{CHash}(pk_{\text{san}}, \mathsf{TAG}_0, (\mathsf{TAG}_1, \cdots, \mathsf{TAG}_\gamma, m), r_i) = \mathsf{CHash}(pk_{\text{san}}, \mathsf{TAG}'_0, (\mathsf{TAG}'_1, \cdots, \mathsf{TAG}'_\gamma, m'), r'_i)$$

Set $TAG_i \leftarrow \mathcal{PRG}(x_i)$, where $x_i \leftarrow \mathcal{PRF}(\kappa, n_i)$. Output π, where

$$\pi = ((\mathsf{TAG}_i)_{0 \le i \le \gamma}, m, pk_{\text{sig}}, pk_{\text{san}}, (r_i)_{0 \le i \le \gamma}, (x_i)_{0 \le i \le \gamma})$$

If any errors occur, output $\perp$. In other words, Proof outputs the original blocks as the proof for the complete message.

g) GJudge*:* On input of m, σ, pk_{sig}, pk_{san}, an index i, and the proof π:

$$\pi = ((\mathsf{TAG}_i^\pi)_{0 \le i \le \gamma}, m^\pi, pk_{\text{sig}}^\pi, pk_{\text{san}}^\pi, (r_i^\pi)_{0 \le i \le \gamma}, (x_i^\pi)_{0 \le i \le \gamma})$$

Then check, if σ verifies. Afterwards, check, if $pk_{\text{san}}^\pi = pk_{\text{san}}$. Else, return $\perp$. Let

$$d_i \leftarrow \begin{cases} \texttt{San} & \text{if the collision is non-trivial and} \\ & \mathsf{TAG}_i^\pi = \mathcal{PRG}(x_i^\pi) \\ \texttt{Sig} & \text{else} \end{cases}$$

If $\mathsf{TAG}_0^\pi \neq \mathcal{PRG}(x_0^\pi)$ and there exists no non-trivial collision for the outer chameleon-hash, set $d_i = \texttt{Sig}$. Output d_i, or $\perp$ on error.

h) Judge*:* On input of m, σ, pk_{sig}, pk_{san} and the proof $\pi = ((\mathsf{TAG}_i^\pi)_{0 \le i \le \gamma}, m^\pi, pk_{\text{sig}}^\pi, pk_{\text{san}}^\pi, (r_i^\pi)_{0 \le i \le \gamma}, (x_i^\pi)_{0 \le i \le \gamma})$ let $d_i \leftarrow \mathsf{GJudge}(m, \sigma, pk_{\text{sig}}, pk_{\text{san}}, \pi, i)$ for each group $\text{GRP}[i] \in \text{GRP}$. If $\mathsf{TAG}_0^\pi \neq \mathcal{PRG}(x_0^\pi)$ and there exists no non-trivial collision for the outer chameleon-hash, output $\texttt{Sig}$. On error, output $\perp$. If $\exists i : d_i \neq \texttt{Sig}$, then output $\texttt{San}$ and $\texttt{Sig}$ otherwise.

Theorem 3 (The Construction is Secure.): If the underlying signature scheme $\mathcal{SS}$ is unforgeable, the used chameleon hash is collision resistant under random tagging attacks, while $\mathcal{PRF}$ and $\mathcal{PRG}$ are pseudorandom, our construction is transparent, private, immutable, group-level accountable and unforgeable.

The proofs are relegated to App. A.

C. Group-level Publicly Accountable SanSig

Next, we present a provably secure construction which is private, immutable, group-level non-interactive publicly accountable and unforgeable based on our first construction. This construction alters our first construction such that it removes transparency, but efficiently gives group-level non-interactive public accountability. We achieve this with a constant number of signatures compared to *Brzuska* et al.'s construction [10] where the number of signatures increases linearly with the number of blocks:

a) KGen_{sig}*:* Generate a key pair of the underlying signature algorithm SKeyGen, i.e., $(pk, sk) \leftarrow \mathsf{SKeyGen}(1^\lambda)$. Output $(pk_{\text{sig}}, sk_{\text{sig}}) = (pk, sk)$.

b) KGen_{san}*:* Generate two key pairs, one for the underlying chameleon hash and one for an unforgeable signature scheme. In particular, let $(pk_{\text{san}}.pk_c, sk_{\text{san}}.sk_c) \leftarrow \mathsf{CHKeyGen}(1^\lambda)$ and $(pk_{\text{san}}.pk_s, sk_{\text{san}}.sk_s) \leftarrow \mathsf{SKeyGen}(1^\lambda)$. Output $(pk_{\text{san}}, sk_{\text{san}}) = ((pk_{\text{san}}.pk_c, pk_{\text{san}}.pk_s), (sk_{\text{san}}.sk_c, sk_{\text{san}}.sk_s))$

c) Sign*:* On input of the message $m = (m[1], \cdots, m[\ell])$, $m[i] \in \{0,1\}^*$, pk_{san}, sk_{sig}, ADM, and GRP, draw γ nonces: $s_i = r_i \leftarrow \{0,1\}^\lambda$ and γ additional tags, i.e., $\mathsf{TAG}_i \leftarrow \{0,1\}^{2\lambda}$ Let:

$$h[i] \leftarrow \mathsf{CHash}(pk_{\text{san}}.pk_c, \mathsf{TAG}_i, (i, \text{GRP}[i]), r_i)$$

for all $i = 1, \cdots, \gamma$. Generate:

$$\sigma_c \leftarrow \mathsf{SSign}(sk_s, (h[1], \cdots, h[\gamma], \text{GRP}[0], pk_{\text{san}}, \text{ADM}, \text{GRP}, (r_i)_{0 < i \le \gamma})$$

and

$$\sigma_d \leftarrow \mathsf{SSign}(sk_s, (h[1], \cdots, h[\gamma], s_1, \cdots, s_\gamma, m))$$

Output:

$$\sigma = (\sigma_c, \sigma_d, (\mathsf{TAG}_i)_{0<i\leq\gamma}, (r_i)_{(0<i\leq\gamma)}, (s_i)_{0<i\leq\gamma}, \text{ADM}, \text{GRP})$$

d) Verify*:* On input of pk_{sig}, pk_{san}, m, $\sigma = (\sigma_c, \sigma_d, (\mathsf{TAG}_i)_{0<i\leq\gamma}, (r_i)_{0<i\leq\gamma}, (s_i)_{0<i\leq\gamma}, \text{ADM})$ compute: $h[i] \leftarrow \mathsf{CHash}(pk_{\text{san}}.pk_c, \mathsf{TAG}_i, (i, \text{GRP}[i]), s_i)$ Check, if σ_d either verifies under $pk_{\text{san}}.pk_s$ or pk_{sig}. If σ_d verifies under pk_{sig}, also check, if the r_i protected by σ_c and σ_d are equal, i.e., if $r_i = s_i$. If so, output:

$$\mathsf{SVerify}(pk, (h[i]_{0<i\leq\gamma}, \text{GRP}[0], pk_{\text{san}}, \text{ADM}, \text{GRP}, (s_i)_{0<i\leq\gamma}), \sigma_c)$$

e) Sanit*:* On input of pk_{sig}, sk_{san}, m, MOD and $\sigma = (\sigma_c, \sigma_d, (\mathsf{TAG}_i)_{0<i\leq\gamma}, (r_i)_{0<i\leq\gamma}, (s_i)_{0<i\leq\gamma}, \text{ADM})$ check, if the received message-signature pair is valid using Verify. If not, stop and output $\perp$. For each **group** $\text{GRP}[i] \in \text{MOD}$, draw new tags $\mathsf{TAG}'_i \leftarrow \{0,1\}^{2\lambda}$. If $\text{GRP}[i] \notin \text{MOD}$, set $\mathsf{TAG}'_i = \mathsf{TAG}_i$ and $s'_i = r_i$. Afterwards, compute:

$$s'_i \leftarrow \mathsf{CHAdapt}(sk_{\text{san}}.sk_c, \mathsf{TAG}_i, \text{GRP}[i], r_i, \mathsf{TAG}'_i, \text{GRP}[i]')$$

Output (m', σ'), where $m' \leftarrow \text{MOD}(m)$ and

$$\sigma' = (\sigma_c, \sigma'_d, (\mathsf{TAG}')_{0<i\leq\gamma}, (r_i)_{0<i\leq\gamma}, (s'_i)_{0<i\leq\gamma}, \text{ADM}, \text{GRP})$$

where $\sigma'_d \leftarrow \mathsf{SSign}(sk_{\text{san}}.sk_s, (h[1], \cdots, h[\gamma], s'_1, \cdots, s'_\gamma, m'))$ Again, we want to emphasize, that $r'_i = r_i$, where $\text{GRP}[i] \in \text{MOD}$, is only possible with negligible probability, if the TAG is changed.

f) Proof*:* Always return $\perp$.

g) GDetect*:* On input of $m = (m[1], \cdots, m[\ell])$, $m[i] \in \{0,1\}^*$, and

$$\sigma = (\sigma_c, \sigma_d, (\mathsf{TAG}_i)_{0<i\leq\gamma}, (r_i)_{0<i\leq\gamma}, (s_i)_{0<i\leq\gamma}, \text{ADM}, \text{GRP}),$$

pk_{sig}, pk_{san}, $\perp$ and an index i, first check, if σ verifies. For group $\text{GRP}[i] \in \text{GRP}$ let:

$$d_i \leftarrow \begin{cases} \texttt{San} & \text{if } r_i \neq s_i \\ \texttt{Sig} & \text{else} \end{cases}$$

Output d_i, or $\perp$ on error resp.

h) Judge*:* On input of m,

$$\sigma = (\sigma_c, \sigma_d, (\mathsf{TAG}_i)_{0<i\leq\gamma}, (r_i)_{0<i\leq\gamma}, (s_i)_{0<i\leq\gamma}, \text{ADM}),$$

pk_{sig}, pk_{san} and $\perp$, first check, if σ verifies. For each $\text{GRP}[i] \in \text{GRP}$, call $d_i \leftarrow \mathsf{GJudge}(m, \sigma, pk_{\text{sig}}, pk_{\text{san}}, \perp, i)$. On error, output $\perp$. If $\exists i : d_i \neq \texttt{Sig}$, then output `San` and `Sig` otherwise.

Theorem 4 (The Construction is Secure.): If the underlying signature scheme $\mathcal{SS}$ is unforgeable, the used chameleon hash is collision resistant under random tagging attacks, while $\mathcal{PRF}$ and $\mathcal{PRG}$ are pseudorandom, our construction is private, immutable, group-level non-interactive publicly accountable and unforgeable.

The proofs are relegated to App. A.

D. Performance Measurements

We have implemented our SanSigs and the construction by *Brzuska* et al. [7]. The source code used for this evaluation will be made available on request. The tests were performed on a *Fujitsu Celsius* with an *Intel* Q9550 Quad Core `@2.83` GHz and 3 GiB of RAM. We only used one core and utilized RSA as the signature algorithm. The moduli have been fixed to 512, 1,024, 2,048 and 4,096-Bit. We evaluated every algorithm with 100, 500 and 1,000 blocks. We fixed the amount of admissible blocks to 50% and always sanitized all admissible blocks. Moreover, to maintain comparability, each group is exactly one block of the message signed, i.e., $\gamma = |\text{ADM}|$. We omit the key pair generation, as we assume that the key pairs are pre-generated. Proof and Judge are very fast, as they contain only a database lookup and are therefore omitted. The results can be seen in Tab. I, and Tab. II, Tab. III.

As seen, the performance is nearly the same for all three schemes. Hence, our constructions are as useable as the one by *Brzuska* et. al [7].

V. Conclusion and Open Questions

In this paper, we have introduced the new notion of group-level properties for sanitizable signatures. In particular, we have formalized the notions of group-level accountability, both in an offline and an online variant. The offline variant allows to achieve transparency which positively answers *Brzuska* et al.'s open question [10]. Our broader scope of groups of blocks in the definitions includes all existing notions of accountability on block- [10] or message-level [7]. Hence, we offer a real generalization, which closes current gaps. We have derived two novel yet provably secure constructions, achieving our new notions. Both constructions show how the group-level definitions allow to choose between performance and accountability: the signer can control the granularity of accountability. The performance analysis highlights that the scheme by *Brzuska* et al. [7] and our two new schemes are reasonable performant, while our construction achieves the stronger security notion of transparency. It remains an open question how to offer the new paradigm of accountability on overlapping groups of blocks, which would allow to capture security concepts like the four-eye principle. Furthermore, it is still unclear how to mix the non-interactive public accountability and the interactive transparent accountability in one message.

References

[1] J. H. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, A. Shelat, and B. Waters. Computing on authenticated data. Cryptology ePrint Archive, Report 2011/096, 2011. http://eprint.iacr.org/.

[2] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik. Sanitizable signatures. In *ESORICS*, pages 159–177. Springer, 2005.

[3] N. Attrapadung, B. Libert, and T. Peters. Computing on authenticated data: New privacy definitions and constructions. In *ASIACRYPT*, pages 367–385, 2012.

[4] N. Attrapadung, B. Libert, and T. Peters. Efficient completely context-hiding quotable and linearly homomorphic signatures. In *PKC*, pages 386–404, 2013.

[5] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *EUROCRYPT*, pages 92–111, 1994.

[6] C. Brzuska, H. Busch, O. Dagdelen, M. Fischlin, M. Franz, S. Katzenbeisser, M. Manulis, C. Onete, A. Peter, B. Poettering, and D. Schröder. Redactable Signatures for Tree-Structured Data: Definitions and Constructions. ACNS'10, pages 87–104. Springer, 2010.

TABLE I. PERFORMANCE OF OUR FIRST SCHEME; MEDIAN RUNTIME IN MS

	Signing			Verifying			Sanitizing		
λ \ ℓ	100	500	1,000	100	500	1,000	100	500	1,000
512 Bit	16	63	125	15	46	78	157	766	1,641
1,024 Bit	28	112	14,132	20	96	22	1,007	4,948	9,720
2,048 Bit	110	391	750	62	328	657	7,109	35,328	70,997
4,096 Bit	563	1,546	2,798	250	1,235	2,469	54,719	272,672	545,062

TABLE II. PERFORMANCE OF OUR SECOND SCHEME; MEDIAN RUNTIME IN MS

	Signing			Verifying			Sanitizing			Detecting		
λ \ ℓ	100	500	1,000	100	500	1,000	100	500	1,000	100	500	1,000
512 Bit	16	78	140	15	47	94	172	797	1,578	16	46	94
1,024 Bit	47	172	313	31	141	265	1,047	5,062	10,438	32	125	266
2,048 Bit	172	516	969	94	437	875	7,547	36,079	72,735	93	421	859
4,096 Bit	922	2,157	4,141	328	1,546	3,546	55,453	271,329	562,683	360	1,546	3,109

TABLE III. PERFORMANCE OF THE SCHEME BY *Brzuska* ET AL. [7]; MEDIAN RUNTIME IN MS

	Signing			Verifying			Sanitizing		
λ \ ℓ	100	500	1,000	100	500	1,000	100	500	1,000
512 Bit	15	46	93	16	31	78	156	781	1,532
1,024 Bit	31	125	219	32	109	203	984	4,875	9,703
2,048 Bit	110	391	765	62	328	672	7,109	34,747	70,782
4,096 Bit	594	1,547	2,750	250	1,250	2,453	57,390	273,625	537,110

[7] C. Brzuska, M. Fischlin, T. Freudenreich, A. Lehmann, M. Page, J. Schelbert, D. Schröder, and F. Volk. Security of Sanitizable Signatures Revisited. In *Proc. of PKC 2009*, pages 317–336. Springer, 2009.

[8] C. Brzuska, M. Fischlin, A. Lehmann, and D. Schröder. Sanitizable signatures: How to partially delegate control for authenticated data. In *Proc. of BIOSIG*, volume 155 of *LNI*, pages 117–128. GI, 2009.

[9] C. Brzuska, M. Fischlin, A. Lehmann, and D. Schroeder. Unlinkability of sanitizable signatures. In *PKC*, pages 444–461, 2010.

[10] C. Brzuska, H. C. Pöhls, and K. Samelin. Non-Interactive Public Accountability for Sanitizable Signatures. In *EuroPKI*, volume 7868 of *LNCS*, pages 178–193. Springer-Verlag, 2012.

[11] S. Canard and A. Jambert. On extended sanitizable signature schemes. In *CT-RSA*, pages 179–194, 2010.

[12] S. Canard, A. Jambert, and R. Lescuyer. Sanitizable signatures with several signers and sanitizers. In *AFRICACRYPT*, pages 35–52, 2012.

[13] S. Canard, F. Laguillaumie, and M. Milhau. Trapdoor sanitizable signatures and their application to content protection. In *ACNS*, pages 258–276, 2008.

[14] E.-C. Chang, C. L. Lim, and J. Xu. Short Redactable Signatures Using Random Trees. In *CT-RSA*, pages 133–147. Springer, 2009.

[15] J. Gong, H. Qian, and Y. Zhou. Fully-secure and practical sanitizable signatures. volume 6584 of *LNCS*, pages 300–317. Springer Berlin / Heidelberg, 2011.

[16] S. Haber, Y. Hatano, Y. Honda, W. G. Horne, K. Miyazaki, T. Sander, S. Tezoku, and D. Yao. Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In *ASIACCS*, pages 353–362, 2008.

[17] R. Johnson, D. Molnar, D. Song, and D.Wagner. Homomorphic signature schemes. In *CT-RSA*, pages 244–262. Springer, Feb. 2002.

[18] M. Klonowski and A. Lauks. Extended Sanitizable Signatures. In *ICISC*, pages 343–355, 2006.

[19] H. Krawczyk and T. Rabin. Chameleon Hashing and Signatures. In *NDSS*, pages 143–154, 2000.

[20] A. Kundu and E. Bertino. Structural Signatures for Tree Data Structures. In *Proc. of PVLDB 2008*, New Zealand, 2008. ACM.

[21] M. Mambo, K. Usuda, and E. Okamoto. Proxy signatures for delegating signing operation. In *CCS*, CCS '96, pages 48–57, New York, NY, USA, 1996. ACM.

[22] K. Miyazaki, M. Iwamura, T. Matsumoto, R. Sasaki, H. Yoshiura, S. Tezuka, and H. Imai. Digitally Signed Document Sanitizing Scheme with Disclosure Condition Control. *IEICE Transactions*, 88-A(1):239–246, 2005.

[23] K. Miyazaki, S. Susaki, M. Iwamura, T. Matsumoto, R. Sasaki, and H. Yoshiura. Digital documents sanitizing problem. Technical report, 2003.

[24] H. C. Pöhls, K. Samelin, and J. Posegga. Sanitizable Signatures in XML Signature - Performance, Mixing Properties, and Revisiting the Property of Transparency. In *ACNS*, volume 6715 of *LNCS*, pages 166–182. Springer-Verlag, 2011.

[25] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 26(1):96–99, 1983.

[26] K. Samelin, H. C. Pöhls, A. Bilzhause, J. Posegga, and H. de Meer. On Structural Signatures for Tree Data Structures. In *ACNS*, volume 7341 of *LNCS*, pages 171–187. Springer-Verlag, 2012.

[27] K. Samelin, H. C. Pöhls, A. Bilzhause, J. Posegga, and H. de Meer. Redactable signatures for independent removal of structure and content. In *ISPEC*, volume 7232 of *LNCS*, pages 17–33. Springer-Verlag, 2012.

[28] R. Steinfeld and L. Bull. Content extraction signatures. In *ICISC 2001*. Springer Berlin / Heidelberg, 2002.

APPENDIX

A. Security of Construction 1.

It is enough to show that the scheme is group-level accountable, transparent and immutable due to the implications given in this work and by *Brzuska* et al. [7], [10]. We prove each property on its own. Most of the proofs are kept short, as they are comparable to the ones given in [7], [10].

Theorem 5 (Construction 1 is Secure.): If the underlying signature scheme $\mathcal{SS}$ is UNF-CMA, while the used chameleon hash is collision-resistant under random-tagging attacks, our construction is transparent, private, immutable, unforgeable and group-level accountable. We prove each property on its own.

Our scheme is immutable: Let $\mathcal{A}$ denote an efficient adversary breaking the immutability of our scheme. We can then construct an adversary $\mathcal{B}$ using $\mathcal{A}$ as a black box to break the unforgeability of the underlying signature scheme as follows. We simulate $\mathcal{A}$'s environment by simulating the signing oracle; the signature of the underlying signature scheme (σ_c)

is generated by $\mathcal{B}$'s own oracle. Eventually, $\mathcal{A}$ will output a forgery attempt, i.e., a tuple (pk^*, m^*, σ^*). This finishes the simulation. We have to distinguish between three cases: (1) We have $pk_{\text{san}} \neq pk_{\text{san},i}$ for all queries. As pk_{san} has been signed, the underlying signature scheme has been broken. (2) For some j and $i_j \notin \text{ADM}_j$, $m_j^*[j_i] \neq m_j[j_i]$ yields. As m^* has therefore not been queried, the unforgeability of the underlying signature scheme has been broken as well. (3) For some group $\text{GRP}_j[i]$, the message has been replaced by a hash or vice versa resp. As this implies $\text{GRP}_j[i] \neq \text{GRP}^*[i]$, the signature must have been forged, as GRP is signed. If neither case happens, the simulation aborts. The signature forgeries can be extracted in all cases and are then returned by $\mathcal{B}$ as a valid forgery of the underlying signature scheme. Hence, $\mathcal{B}$'s success probability *equals* the one of $\mathcal{A}$. ∎

Our scheme is transparent: Transparency follows from the definitions of CHash and CHAdapt, as the distribution of r' and h are computationally indistinguishable from uniform [7]. Moreover, the pseudorandom generators output numbers which are computationally indistinguishable from uniform as well. We do not consider any tag-collisions here, as they only appear with negligible probability. Transparency follows. ∎

Our scheme is group-level sanitizer accountable: Please note, in the case where $h[i] \neq h^*[i]$, where h denotes the digest of a group, a direct forgery of the underlying signature scheme is implied. This is also true for $pk_{\text{san},i} \neq pk^*$ and $\text{ADM}_i \neq \text{ADM}^*$ and $\text{GRP}_i \neq \text{GRP}^*$. Also note, that in this case the Proof-oracle can trivially be simulated by picking κ itself. Hence, we can focus on the chameleon hash. To be successful, the adversary against group-level signer accountability needs to make sure that the proof algorithm Proof cannot find at least one non-trivial colliding pair of chameleon hash digests. Hence, we have:

$$\mathsf{CHash}(pk_{\text{san}}, \mathsf{TAG}_{j,0}, ((\mathsf{TAG}_{j,i})_{0\leq i\leq\gamma}, m), r_{j,0}) = \mathsf{CHash}(pk^*, \mathsf{TAG}^*_{j,0}, ((\mathsf{TAG}_{j,i})^*_{0\leq i\leq\gamma}, m^*), r_0^*)$$

for some query j. However, this collision is non-trivial and Proof can find it, which prohibits the attack. This also applies to the outer chameleon hash, protecting against match-and-mix attacks. Building an extractor is straight forward and therefore omitted. Sanitizer accountability for groups follows. ∎

Our scheme is group-level signer accountable: Let $\mathcal{A}$ denote an efficient adversary breaking the group-level signer accountability of our scheme. We can then construct an adversary $\mathcal{B}$ using $\mathcal{A}$ as a black box to break the collision-resistance against random-tagging attacks of the underlying chameleon hash in the follow way. As before, $\mathcal{B}$ simulates $\mathcal{A}$'s environment. However, calls to the sanitization oracle are simulated using $\mathcal{B}$'s $OAdapt$-oracle and signed by its own generated signature key pair. Eventually, $\mathcal{A}$ returns $(pk^*, \pi^*, m^*, \sigma^*)$.

By definition π^* must contain two (non-trivial) colliding tuples:

$$\mathsf{CHash}(pk_{\text{san}}, \mathsf{TAG}_{j,0}, (\mathsf{TAG}_{j,i})_{0\leq i\leq\gamma}, r_{j,i}) = \mathsf{CHash}(pk^*, \mathsf{TAG}^*_i, (\mathsf{TAG}_{j,i})_{0\leq i\leq\gamma})^*, r_i^*)$$

This finishes the simulation. Afterwards, $\mathcal{B}$ outputs the colliding tuples. These tuples break the collision-resistance of the chameleon hash as the tags are drawn at random. Any tag-collision is therefore only possible with negligible probability. Hence, $\mathcal{B}$'s success probability equals the one of $\mathcal{A}$. Please note that this also applies for the outer chameleon hash, protecting against match-and-mix attacks. Building an extractor is straight forward and therefore omitted. Hence, the attack discovered by *Gong* et al. does not apply here, as we add an additional chameleon hash, protecting the whole message, similar to [15]. Signer accountability for groups follows. ∎

B. Security of Construction 2.

Theorem 6 (Construction 2 is Secure.): If the underlying signature scheme $\mathcal{SS}$ is UNF-CMA, while the used chameleon hash is collision-resistant under random-tagging attacks, our construction is private, immutable, unforgeable and group-level non-interactive publicly accountable. Following our definitions and [7], [10], it is enough to show that privacy, immutability and group-level non-interactive public accountability hold to prove the security of our scheme.

Proof: The proofs for privacy, immutability and unforgeability are exactly the same as for our first construction, with two notable exceptions: We do not achieve transparency, as we sign the original $r[i]$. However, the randomness does not leak any information about the original message, as the tags are drawn at random. Moreover, the "outer" signature protects against mix-and-match attacks. In other words, the sanitizer is only able to draw a new tag, which changes the random coin, but not the message, while the random coins for the chameleon hash are always distributed uniformly, which implies privacy.

Therefore, we only need to show that our scheme is group-level non-interactive publicly accountable. Assume that there is an efficient adversary $\mathcal{A}$ against group-level non-interactive public accountability. We can then construct an adversary $\mathcal{B}$ using $\mathcal{A}$ as a black box to break the unforgeability of the underlying signature scheme as follows: $\mathcal{B}$ forwards any queries to its own oracles and returns the answers to $\mathcal{A}$. $\mathcal{B}$ also flips a coin $b \leftarrow \{0, 1\}$. Eventually, $\mathcal{A}$ returns a tuple (pk^*, m^*, σ^*). If $b = 1$, $\mathcal{B}$ sets $pk_{\text{san}} \leftarrow pk^*$ and $(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}$ else, $\mathcal{A}$ sets $pk_{\text{sig}} \leftarrow pk^*$ and $(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}$.

Consequently, we have to distinguish between two cases, i.e., a malicious sanitizer and a malicious signer. The probability that the simulation is done for the correct case is exactly $\frac{1}{2}$. We will omit cases where the random coins are equal, as this only occurs with negligible probability.

a) Malicious Signer: As $r'_i \neq r_i$, the underlying signature scheme must been forged, as σ_d protects all r_i, as $r'_i = r_i$ occurs only with negligible probability.

b) Malicious Sanitizer: We know that $r'_i = r_i$ only occurs with negligible probability. Therefore, σ_d must be a valid forgery.

In both cases, an extractor can trivially be build. ∎

ACKNOWLEDGEMENTS

Henrich C. Pöhls was funded by BMBF (FKZ:13N10966) and ANR as part of the ReSCUeIT project. Kai Samelin was supported by "Regionale Wettbewerbsfähigkeit und Beschäftigung", Bayern, 2007-2013 (EFRE) as part of the SECBIT project (http://www.secbit.de) and the European Community's Seventh Framework Programme through the EINS Network of Excellence (grant agreement no. [288021]).