Non-Interactive Public Accountability for Sanitizable Signatures

Christina Brzuska[1], Henrich C. Pöhls[2,4]*, Kai Samelin[3,4]**

[1] Darmstadt University of Technology, Germany & CASED
[2] Chair of IT-Security
[3] Chair of Computer Networks and Computer Communication
[4] Institute of IT-Security and Security Law (ISL), University of Passau, Germany
`brzuska@cased.de, {hp,ks}@sec.uni-passau.de`

Abstract. Sanitizable signatures enable a designated party to modify signed documents in a controlled way, while the derived signature still verifies. In this paper, we introduce the notion of *non-interactive* and *public* accountability. It allows a third party to determine whether a message-signature pair was issued by the signer or the sanitizer. The original notion of accountability does not satisfy European legal standards, while non-interactive public accountability does. A contradictory security goal is the indistinguishability of message-signature pairs from the signer and the sanitizer, a.k.a. transparency. As state-of-the-art schemes often satisfy transparency, they can only achieve a weaker notion of accountability. We show that non-interactive public accountability does not contradict privacy by proving that an existing scheme by Brzuska et al. (BIOSIG '09) satisfies both notions. We then extend the scheme to also satisfy *blockwise* public accountability. Overall, for e-business applications within the EU, opting for non-interactive public accountability can be preferable over transparency.

Keywords: Accountability, Sanitizable Signatures, Privacy, e-commerce

1 Introduction

Digital signatures protect the integrity and authenticity of data. In many settings, digital signatures serve as a substitute for handwritten signatures on paper. If they achieve the legal requirements, they are even recognized as equivalent [12]. Cryptographically, digital signatures detect malicious or accidental subsequent changes and identify the signer by its public key. Hence, the verification of a digital signature must fail, whenever the signed data was altered.

*Is funded by BMBF (FKZ:13N10966) and ANR as part of the ReSCUeIT project
**The research leading to these results 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]).

Legal requirements on digital signatures vary upon different countries. We focus on the European Union (EU) directive for electronic signatures, as it covers the entire EU and is technology neutral [12]. From a legal perspective, also non-cryptographic solutions such as scanned handwritten signatures are considered electronic signatures. Laborde points out, that the directive [12] "implicitly" [16] endorses cryptographic digital signatures, such as RSA-PSS, for so-called *advanced* electronic signatures by "establishing requirements that, so far, can only be fulfilled by using digital signatures" [16]. If public keys are certified and other technical and organizational requirements set forth by law are met, the signatures created by advanced signature schemes legally become *advanced electronic signatures based on qualified certificates* and are granted *Statutory Trust* by the EU electronic signature laws [12]. In a nutshell, only then they are "legally equivalent to hand-written signatures". Documents signed with aforementioned signatures offer legally a high value of evidence and are hence economically valuable.

Contrary, applications such as secure routing or anonymization of medical data [2] require controlled modification of signed data. Moreover, this type of modification must not require any interaction with the signer, as this might be inconvenient, impossible or legally forbidden. Miyazaki et al. [18] called this constellation the "digital document sanitization problem". An elegant cryptographic solution are sanitizable signatures (SanSig) [2]. In a SanSig, the signer signs a message m that is split into ℓ blocks, i.e., $m = (m[1], m[2], \ldots, m[\ell])$. For each block $m[i]$, the signer decides whether sanitization is admissible or not. A designated party, called *sanitizer*, can replace each admissible block $m[i]$ with an arbitrary string $m[i]' \in \{0,1\}^*$ and create a new message m'. For m', the sanitizer derives a new signature σ', without interaction with the signer, so that (m', σ') verifies.

Cryptographic Transparency versus Legal Requirements. The cryptographic security property of transparency says that it is impossible for third parties to decide which party is accountable for a given signature-message pair. We want to emphasize that the word "transparency" is not used in the legal sense, but to describe a cryptographic property[1]: the sanitizer remains *anonymous* to the verifier. In other words, a verifier of a signature-message pair is not able to decide, if the sanitizer has used the sanitizer secret key to generate a new signature over the potentially sanitized message, or, if the message still bears the original signature and was not touched. A related property is privacy: a private SanSig hides all information about the original message. Transparency hides the involvement of the sanitizer from the verifier and is desirable in cases, where already the fact that sanitization has occurred leads to discrimination.

We now sketch how cryptographic transparency conflicts legal requirements and point out that, fortunately, transparency is not needed in all practical applications of SanSigs. Assume a hospital's accountant creates a bill for the patient's

[1] This nomenclature is in line with state-of-the-art work in SanSig.

insurance company. Accountancy only needs some information. Thus, the hospital can automatically omit or de-identify other medical data before passing the record to the accountant. Such a sanitized patient record contains empty medical details which indicates sanitization. A transparent SanSig will try to prevent the accountant from identifying the record as sanitized from only looking at the signature and record. However, the accountant notices obvious sanitizations through domain specific information, like omissions of names, anyway. So transparency is neither required nor used in this scenario. Still, privacy is required, i.e., the patient's sanitized medical data must not be recoverable by the accountant. Second, transparency diminishes the legal value of the signature. Unfortunately, as observed by Pöhls and Höhne [19], transparent SanSig do not fulfill the strict legal requirements. Only if a SanSig allows detecting all changes instantly and allows holding a single person accountable, the SanSig can offer a comparable legal value of evidence as advanced digital signatures.

1.1 Non-Interactive Public Accountability

Accountability is required by all SanSig, and is independent of transparency [5]. By the very definition of transparency, transparent schemes cannot achieve non-interactive public accountability. Instead, they achieve a weaker notion of *interactive accountability*, which requires the signer's secret key to run the additional algorithm Proof. If the signer does not participate in the protocol run, it is declared accountable for the signature by Judge. In contrast, standard signature schemes require no interaction to determine the signature's origin. Verification can be done non-interactively with the knowledge of the signer's public key.

We show in Theorem 2 that it is insufficient to negate the property of transparency, e.g., when one is able to correctly detect $\frac{3}{4}$ of all sanitizer involvements, the scheme is neither accountable, nor transparent. Moreover, a lack of transparency does not guarantee security against malicious signers/sanitizers. Hence, we require a mechanism to allow a third party to decide whether a sanitizer's secret key was involved in the signature generation or if only the signer's secret key was involved even in the presence of malicious signers/sanitizers. The key's involvement can then be linked to the corresponding parties and determine the origin of the signature. This leads to our definition:

> A sanitizable signature scheme satisfies *non-interactive public accountability*, if and only if for a valid message/signature pair (m, σ), a third party can correctly decide whether (m, σ) originates from the signer or from the sanitizer without interacting with the signer or sanitizer.

Sometimes, it is not sufficient to only decide which party is accountable for the entire signature-message pair. Recall the above example: non-interactive public accountability detects that the medical record itself has changed — however, if more than one entry of the medical record, each represented by one block, is modifiable, a third party may need to know which blocks have undergone a

sanitization. Hence, to filter potentially sanitized from unchanged information, we introduce accountability for *each* block $m[i]$ of the received message. Consider the following clarifying example: when medical records are anonymized to pass them to the World Health Organization (WHO), the sanitizer is allowed to replace names (and further identifying information). For de-identification [10], the sanitizer replaces them with zeroes. Thus, generally, the sanitizer is allowed to modify the name field. Now, a malicious sanitizer can exchange two patients' medical records by swapping their names by sanitizing both medical records. Damage is done when the swapped medical records in the hospital's database are consulted for the patients' treatment. The accountability of existing transparent SanSig allows discovering interactively with the signer that this attack has taken place. Unfortunately, the real-life consequences may have occurred already. When sanitization is allowed on multiple blocks knowing that some of these blocks are still unchanged is also legally important. Our new blockwise non-interactive public accountability property allows this:

> A SanSig offers *blockwise public accountability*, if and only if for all valid pairs (m, σ) and their blocks $m[i]$, a third party can always correctly decide if the given block/signature pair $(m[i], \sigma)$ originates from the signer or from the sanitizer.

1.2 State of the Art

Standard security properties of SanSigs were first introduced by Ateniese et al. [2] and later formalized [5]. Brzuska et al. [7] introduce the concept of unlinkability, a strong privacy notation which prohibits a third party from linking two messages. Current techniques to assure computational notions of unlinkability require the expensive use of group signatures [7] — known schemes for statistical notions of unlinkability achieve the less common notion of selective unforgeability [1]. We thus focus on the security properties presented in [5]. In particular, unlike Canard et al. [8, 9], the admissible blocks are an input to the signing algorithm. Moreover, we focus on a setting of a single signer and a single sanitizer. Extending our scheme to multiple sanitizers is straight forward as we do not need to fulfill transparency. Note, unlinkability and public accountability are not mutually exclusive. All known unlinkable schemes are also transparent [1, 7, 8].

A related concept are redactable signatures, introduced in [14, 22]. In these malleable signature schemes, blocks cannot be modified; they can only be removed. Security models and redactable signature schemes have been studied. Exemplary, refer to [4, 11, 15, 17, 20, 21, 23]. Approaches to combine redactable signature schemes and SanSigs appeared in [13]. In this paper, we focus on SanSigs, as redactable signatures schemes have a different goal: They allow public redactions and the remaining blocks are always unchanged and, hence the signer is accountable for the remaining data.

Brzuska et al. [6] provide an elegant construction that is provably secure and in particular achieves privacy, but not transparency. We use their construction as it can be used with existing standard RSA-based signature algorithms and the signers and sanitizers public keys can be bound to identities by public-key certificates. We show that their scheme achieves public accountability on the message level, but it has not been mentioned as a separate property. We extend their construction to also achieve blockwise public accountability.

1.3 Contribution and Outline

We formalize non-interactive public accountability on the message level as well as blockwise in order to satisfy legal requirements for advanced electronic signatures. We show that while non-interactive public accountability and transparency are mutually exclusive, we do not gain anything from the mere absence of transparency. We prove that the non-transparent scheme from [6] achieves non-interactive public accountability and extend it to also provide blockwise non-interactive public accountability. Moreover, the constructions fulfill standard privacy and unforgeability requirements for sanitizable signatures as formalized by Brzuska et al. [5]. Our detailed performance analysis shows that both constructions are usable in practice. Moreover, they fit well into the current public key infrastructure when using a standard signature scheme and certified public keys for signers and sanitizers.

1.4 Open Questions

We introduce public accountability and extended the notion to its blockwise equivalent. For many features of malleable signature schemes, it remains desirable to formalize the corresponding blockwise notion and provide provably secure constructions. Moreover, extending the scheme to groups of blocks might yield additional efficiency gains. Finally, it remains open research to show, if and how unlinkable schemes with non-interactive public accountability can be constructed.

2 Notations and Preliminaries

In this section, we shortly revisit the required algorithms and notation. We adapt the nomenclature from [5]. For a message $m = (m[1], \ldots, m[\ell])$, we call $m[i]$ a *block*. , denotes a uniquely reversible concatenation, while $\perp \notin \{0,1\}^*$ denotes a special symbol not contained in messages to indicate an error.

Definition 1 (Sanitizable Signature Scheme). *Any* SanSig *consists of seven PPT algorithms* (KGen_{sig}, KGen_{san}, Sign, Sanit, Verify, Proof, Judge)*, such that:*

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 key and the corresponding public key:*

$$(\mathrm{pk}_{sig}, \mathrm{sk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\lambda), \qquad (\mathrm{pk}_{san}, \mathrm{sk}_{san}) \leftarrow \mathsf{KGen}_{san}(1^\lambda)$$

Signing. *The* Sign *algorithm takes as input the security parameter* 1^λ*, a message* $m \in \{0,1\}^*$*, the signer's secret key* sk_{sig}*, the sanitizer's public key* pk_{san}*, as well as a description* ADM *of the admissibly modifiable blocks.* ADM *is a set containing just those blocks' indices which are modifiable by the sanitizer. It outputs a signature* σ *and the message (or* $\perp$ *on error):*

$$(m, \sigma) \leftarrow \mathsf{Sign}(1^\lambda, m, \mathrm{sk}_{sig}, \mathrm{pk}_{san}, \mathrm{ADM})$$

We assume that ADM *is always recoverable from any signature* $\sigma \neq \perp$ *by the sanitizer holding the secret key* sk_{san} *that corresponds to* pk_{san}*.*

Sanitizing. *Algorithm* Sanit *takes the security parameter, a message* $m \in \{0,1\}^*$*, a modification instruction* MOD*, a signature* σ*, the signer's public key* pk_{sig} *and the sanitizer's secret key* sk_{san}*. It modifies the message* m *according to the modification instruction* MOD*, which contains pairs* $(i, m[i]')$ *for those blocks that shall be modified.* Sanit *generates a new signature* σ' *for the modified message* $m' \leftarrow \mathrm{MOD}(m)$*. Then* Sanit *outputs* m' *and* σ' *(or* $\perp$ *in case of an error).*

$$(m', \sigma') \leftarrow \mathsf{Sanit}(1^\lambda, m, \mathrm{MOD}, \sigma, \mathrm{pk}_{sig}, \mathrm{sk}_{san})$$

Verification. *The* Verify *algorithm outputs a bit* $d \in \{\mathtt{true}, \mathtt{false}\}$ *indicating the correctness of the signature* σ *for the message* m *with respect to the public keys* pk_{sig} *and* pk_{san}*. The security parameter is denoted by* 1^λ*.*

$$d \leftarrow \mathsf{Verify}(1^\lambda, m, \sigma, \mathrm{pk}_{sig}, \mathrm{pk}_{san})$$

Proof. *The* Proof *algorithm takes as input the security parameter, the signer's secret key* sk_{sig}*, a message* m *and a signature* σ *as well a set of (polynomially many) additional message/signature pairs* $(m_i, \sigma_i)_{i=1,2,\ldots,k}$ *and the public key* pk_{san}*. It outputs a string* $\pi \in \{0,1\}^*$*:*

$$\pi \leftarrow \mathsf{Proof}(1^\lambda, \mathrm{sk}_{sig}, m, \sigma, (m_1, \sigma_1), \ldots, (m_k, \sigma_k), \mathrm{pk}_{san})$$

Judge. *Algorithm* Judge *takes as input the security parameter, a message* m *and a valid signature* σ*, the public keys of the parties and a proof* π*. It outputs a decision* $d \in \{\mathtt{Sig}, \mathtt{San}\}$ *indicating whether the message/signature pair has been created by the signer or the sanitizer (or* $\perp$ *in case of an error):*

$$d \leftarrow \mathsf{Judge}(1^\lambda, m, \sigma, \mathrm{pk}_{sig}, \mathrm{pk}_{san}, \pi)$$

We require the usual correctness properties to hold. In particular, every genuinely signed or sanitized message is accepted and genuinely created proofs by the signer lead the judge to decide in favor of the signer. See [5] for a formal definition of correctness. We emphasize that MOD does not necessarily contain the instruction to modify all blocks. Moreover, if $(i, m[i]') \in \text{MOD}$ it might still be that $m[i]' = m[i]$. Jumping ahead, this difference is important for non-interactive public accountability: it allows a sanitizer to be accountable for a block $m[i]$ without modifying it.

We define a SanSig with non-interactive public accountability to have an empty Proof algorithm. This requires that Judge always correctly decides upon an empty proof ($\pi = \perp$). This simplifies the description, as Proof requires the signer's private key sk_{sig}. We use the notation $m[\text{ADM}]$ for the set of the admissible blocks and $m[\text{FIX}]$ for the set of the fixed blocks. Ateniese et al. introduced a set of desirable properties [2], later rigorously formalized by Brzuska et al. [5]. We shortly re-state them:

- *Immutability* prevents the sanitizer from modifying non-admissible blocks.
- *Unforgeability* assures that third parties cannot produce a signature for a message not issued by the signer nor by the sanitizer. This is similar to the unforgeability requirements of standard signature schemes.
- *Privacy* and its extension *unlinkability* [7] inhibit a third party from learning anything about the original message. For example, from a sanitized medical record, one cannot retrieve any additional information besides what is given.
- *Transparency* says that it should be computationally hard for third parties to decide which party is accountable for a given signature-message pair.
- *Accountability* allows a judge to settle disputes over the origin of a message. The judge may request additional information.

Non-interactive public accountability negatively impacts only on transparency. The definition of transparency given by Brzuska et al. [5] is as follows:

Definition 2 (Transparency). *A sanitizable signature scheme SanSig is* proof-restricted transparent *if for any efficient algorithm $\mathcal{A}$ the probability that the experiment* $\mathsf{Transparency}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ *given in Fig. 1 returns* 1 *is negligibly close to $\frac{1}{2}$ (as a function of λ). The idea is that the adversary must decide if a given message/signature pair has been created by Sign or Sanit.*

By this very definition, transparency prohibits any public form of accountability, which requires that third parties can immediately decide whether a message has been issued by the signer or the sanitizer. Transparency is desirable in some cases, but not essentially required in all constellations. It becomes even obsolete when sanitization is detectable solely from obvious changes to the message. On the contrary, transparency prevents certain usages, as it reduces the legal

Experiment $\mathsf{Transparency}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$b \leftarrow \{0,1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,sk_{\text{sig}},\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,sk_{\text{san}}),\mathsf{Proof}(sk_{\text{sig}},\dots),\mathsf{Sanit/Sign}(\cdot,\cdot,\cdot,sk_{\text{sig}},sk_{\text{san}},pk_{\text{sig}},pk_{\text{san}},b)}(pk_{\text{sig}}, pk_{\text{san}})$
where oracle $\mathsf{Sanit/Sign}$ for input $m_k, \text{MOD}_k, \text{ADM}_k$
if $\text{MOD}_i \not\subseteq \text{ADM}$, return $\perp$
if $b = 0$: first compute $\sigma_k \leftarrow \mathsf{Sign}(1^\lambda, m_k, sk_{\text{sig}}, pk_{\text{san}}, \text{ADM}_k)$,
then compute $(m'_k, \sigma'_k) \leftarrow \mathsf{Sanit}(1^\lambda, m_k, \text{MOD}_k, \sigma_k, pk_{\text{sig}}, sk_{\text{san}})$,
if $b = 1$: first compute $m'_k \leftarrow \text{MOD}_k(m_k)$,
then compute $\sigma'_k \leftarrow \mathsf{Sign}(1^\lambda, m'_k, sk_{\text{sig}}, pk_{\text{san}}, \text{ADM}_k)$,
finally return (m'_k, σ'_k).
return 1 if $a = b$ and $\mathcal{A}$ has not queried any (m_k, σ_k) output by $\mathsf{Sanit/Sign}$ to Proof.

Fig. 1. Security Game for Transparency

value of evidence. Hence, in these cases non-interactive public accountability is preferable.

3 Non-Interactive Public Accountability

In this section, we introduce two variants of non-interactive public accountability. First, we give the formal definition of message level public accountability and second, we provide the formal definition of blockwise public accountability.

3.1 Non-Interactive Public Accountability on Message Level

Our definition of public accountability is derived from the two accountability definitions from Brzuska et al. [5]. In particular, Brzuska et al. considers sanitizer accountability and signer accountability separately. We have merged both definitions by defining that the Proof algorithm always returns $\perp$. Brzuska et al.'s accountability requires the signer's participation to generate the proof π using its private key sk_{sig}. In our case, the signer must not be involved, as Judge is correctly deciding on an empty proof π. In particular, Judge ignores the proof π. Hence, Judge becomes a public and non-interactive algorithm.

Definition 3 (Non-Interactive Public Accountability). *A sanitizable signature scheme SanSig is* non-interactive publicly accountable*, if Proof* $= \perp$ *and if for any efficient algorithm* $\mathcal{A}$ *the probability that the experiment* $\mathsf{Pubaccountability}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ *given in Fig. 2 returns* 1 *is negligible (as a function of* λ*). The basic idea is that an adversary, i.e., the sanitizer or the signer, has to be able to make the Judge decide wrongly on an empty proof* π*.*

By inspection of the definitions, we obtain the following theorem:

Theorem 1. *Let* $\mathsf{SanSig} = (\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$ *with* $\mathsf{Proof} = \perp$*, then SanSig is accountable, if SanSig is publicly accountable.*

Experiment $\mathsf{Pubaccountability}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$(pk^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,sk_{\text{sig}},\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,sk_{\text{san}})}(pk_{\text{sig}}, pk_{\text{san}})$
Let $(m_i, \text{ADM}_i, pk_{\text{san},i})$ and σ_i for $i = 1, 2, \ldots, k$
be the queries and answers to and from oracle Sign
return 1 if
$(pk^*, m^*) \neq (pk_{\text{san},i}, m_i)$ for all $i = 1, 2, \ldots, k$, and
$\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk_{\text{sig}}, pk^*) = \texttt{true}$, and
$\mathsf{Judge}(1^\lambda, m^*, \sigma^*, pk_{\text{sig}}, pk^*, \bot) = \texttt{Sig}$
Let $(m_j, \text{MOD}_j, \sigma_j, pk_{\text{sig},j})$ and (m'_j, σ'_j) for $j = 1, 2, \ldots, k'$
be the queries and answers to and from oracle Sanit
return 1 if
$(pk^*, m^*) \neq (pk_{\text{sig},j}, m'_j)$ for all $j = 1, 2, \ldots, k'$, and
$\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\text{san}}) = \texttt{true}$, and
$\mathsf{Judge}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\text{san}}, \bot) = \texttt{San}$
return 0.

Fig. 2. Public Accountability

Experiment $\mathsf{Blockpubaccountability}^{\mathsf{SanSig}}_{\mathcal{A}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$(pk^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,sk_{\text{sig}},\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,sk_{\text{san}})}$
Let $(m_i, \text{ADM}_i, pk_{\text{san},i})$ and σ_i for $i = 1, 2, \ldots, k$
be the queries and answers to and from oracle Sign
return 1 if
$\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk_{\text{sig}}, pk^*) = \texttt{true}$, and
for all m_i with $pk_{\text{san},i} = pk^*$, there exists a q s.t.
$\mathsf{Detect}(1^\lambda, m^*, \sigma^*, pk_{\text{sig}}, pk^*, q) = \texttt{Sig}$
and $(q, m_i[q]) \in \text{MOD}_i$.
Let $(m_j, \text{MOD}_j, \sigma_j, pk_{\text{sig},j})$, (m'_j, σ'_j) for $j = 1, \ldots, k'$
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
for all m_j with $pk_{\text{sig},j} = pk^*$, there exists a q s.t.
$\mathsf{Detect}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\text{san}}, q) = \texttt{San}$
and $(q, m'_j[q]) \notin \text{MOD}_j$
return 0.

Fig. 3. Blockwise Public Accountability

Proof. In an accountable SanSig, invocation of Judge is public, but needs the proof π generated using secret key sk_{sig} as input to Proof. A correctly working Judge gives the correct answer with overwhelming probability on input of a correct proof π. In a non-interactive public accountability scheme, the invocation of Proof outputs $\bot$. Nonetheless, the public Judge gives the correct answer with overwhelming probability on the input of $\pi = \bot$. Iff the non-interactive public accountability scheme's Judge is correct, the scheme is therefore still accountable.

Intuitively, one might expect that in the absence of transparency it is easy to determine which party issued a message/signature pair. This line of reasoning is, however, incorrect, as the following theorem shows.

Theorem 2. *There exist a SanSig which achieves neither transparency nor non-interactive public accountability.*

Proof. We prove the theorem by giving a concrete example. Let SanSig_1 be a transparent SanSig. We alter the algorithms as follows: when signing, the signer appends a 0 to the signature. The sanitizer, when sanitizing, appends a 1 to the signature. Clearly, signatures from honest signers can be easily distinguished from signatures of honest sanitizers. However, a malicious sanitizer (signer) could append 0 instead of 1 (1 instead of 0) and thus, a third party fails to correctly determine the origin of a signature without executing Proof and Judge.

To separate the two notions even for honest signers and sanitizers, we can alter the above example to having the sanitizer append a random bit. Now, determining the correct origin of a sanitizer's signature succeeds with probability $\frac{1}{2}$, which breaks transparency, that requires it to be negligibly close to 0. Also, the probability of $\frac{1}{2}$ does not satisfy public accountability, that requires it to be negligibly close to 1.

3.2 Blockwise Non-Interactive Public Accountability

Theorem 3. *Every blockwise non-interactive publicly accountable sanitizable signature scheme is also publicly accountable. All other properties are preserved.*

Proof. (sketch) One sanitized block is enough to hold the sanitizer accountable for the complete message. Hence, we define Judge to indicate the sanitizer's involvement, when detecting at least one sanitized block; and to indicate the signer's sole involvement otherwise.

For the following formal definition of blockwise public accountability, we introduce the algorithm Detect. It takes as an input the security parameter, a message m, a valid signature σ as well as the sanitizer's public key pk_{san} and the signer's public key pk_{sig}. Most notably, it also takes as an input a block index i and then returns `San` or `Sig`, indicating which party signed the ith block.

Detect. On input of the security parameter 1^λ, a valid message/signature pair (m, σ), the corresponding public keys pk_{sig} and pk_{san}, and the block index i, Detect outputs the accountable party (`San` or `Sig`) for block i.

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

Definition 4 (Blockwise Non-Interactive Public Accountability). *A sanitizable signature scheme SanSig together with an algorithm Detect is* publicly accountable*, if $\mathsf{Proof} = \perp$ and if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Blockpubaccountability}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ given in Fig. 3 returns 1 is negligible (as a function of λ). The basic idea is that an adversary, i.e., the sanitizer or the signer, has to be able to make the Judge decide wrongly on any block given an empty proof.*

4 Constructions

The first construction is non-interactive publicly accountable, while the second one achieves blockwise non-interactive public accountability. Note, both constructions still satisfy privacy, immutability and unforgeability.

4.1 Construction 1: Non-Interactive Public Accountability

Brzuska et al. [6] give a construction that achieves all properties defined in [5] except for transparency. Building on Th. 1, we conclude that their construction also features non-interactive public accountability. The main construction idea is that the signer signs the fixed parts of the message and additionally, the whole message is signed by the signer or the sanitizer, see Fig. 4 for an overview.

Construction 1 (Non-Interactive Publicly Accountable SanSig) *Let* $\mathsf{S} = (\mathsf{SKGen}, \mathsf{SSign}, \mathsf{SVerify})$ *be an unforgeable signature scheme. Define the sanitizable signature scheme* $\mathsf{SanSig} = (\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Judge})$ *as follows:*

Key Generation: Algorithm KGen_{sig} *generates on input* 1^λ *a key pair* $(\mathrm{pk}_{sig}, \mathrm{sk}_{sig}) \leftarrow \mathsf{SKGen}(1^\lambda)$ *of the underlying signature scheme* S*, and algorithm* KGen_{san} *for input* 1^λ *analogously returns a pair* $(\mathrm{pk}_{san}, \mathrm{sk}_{san}) \leftarrow \mathsf{SKGen}(1^\lambda)$.

Signing: Algorithm Sign *on input* $m \in \{0,1\}^*, \mathrm{sk}_{sig}, \mathrm{pk}_{san}, \mathrm{ADM}$ *computes*

$$\sigma_{\mathrm{FIX}} \leftarrow \mathsf{SSign}(\mathrm{sk}_{sig}, (0, m[\mathrm{FIX}], \mathrm{ADM}, \mathrm{pk}_{san})),$$

$$\sigma_{\mathrm{FULL}} \leftarrow \mathsf{SSign}(\mathrm{sk}_{sig}, (1, m, \mathrm{pk}_{san}, \mathrm{pk}_{sig}))$$

using the underlying signing algorithm. It returns $(m, \sigma) = (m, (\sigma_{\mathrm{FIX}}, \sigma_{\mathrm{FULL}}, \mathrm{ADM}))$.

Sanitizing: Algorithm Sanit *on input message* m*, modification instructions* MOD*, a signature* $\sigma = (\sigma_{\mathrm{FIX}}, \sigma_{\mathrm{FULL}}, \mathrm{ADM})$*, keys* pk_{sig} *and* sk_{san} *first checks that* MOD *is admissible according to* ADM *and that* σ_{FIX} *is a valid signature for message* $(0, m[\mathrm{FIX}], \mathrm{ADM}, \mathrm{pk}_{san})$ *under key* pk_{sig}*. If not, it stops outputting* $\perp$*. If no error occurred, it generates the modified message* $m' \leftarrow \mathrm{MOD}(m)$ *and computes*

$$\sigma'_{\mathrm{FULL}} \leftarrow \mathsf{SSign}(\mathrm{sk}_{san}, (1, m', \mathrm{pk}_{san}, \mathrm{pk}_{sig}))$$

and outputs $(m', \sigma') = (m', (\sigma_{\mathrm{FIX}}, \sigma'_{\mathrm{FULL}}, \mathrm{ADM}))$.

Verification: Algorithm Verify *on input a message* $m \in \{0,1\}^*$*, a signature* $\sigma = (\sigma_{\mathrm{FIX}}, \sigma_{\mathrm{FULL}}, \mathrm{ADM})$ *and public keys* pk_{sig}*,* pk_{san} *first checks that* σ_{FIX} *is a valid signature for message* $(0, m[\mathrm{FIX}], \mathrm{ADM}, \mathrm{pk}_{san})$ *under key* pk_{sig}*. Return* 1*, if either* $\mathsf{SVerify}(\mathrm{pk}_{sig}, (1, m, \mathrm{pk}_{san}, \mathrm{pk}_{sig}), \sigma_{\mathrm{FULL}})$ *or* $\mathsf{SVerify}(\mathrm{pk}_{san}, (1, m, \mathrm{pk}_{san}, \mathrm{pk}_{sig}), \sigma_{\mathrm{FULL}})$ *verifies. If so, it outputs* 1*, declaring the entire signature as valid. Otherwise it returns* 0.

Proof: The Proof *algorithm always returns* $\perp$

Judge: Judge *on input* $m, \sigma, \mathrm{pk}_{sig}, \mathrm{pk}_{san}$ *and* $\perp$ *parses* σ *as* $(\sigma_{\mathrm{FIX}}, \sigma_{\mathrm{FULL}}, \mathrm{ADM})$ *and outputs* $\mathtt{Sig}$ *if* $\mathsf{SVerify}(\mathrm{pk}_{sig}, (1, m, \mathrm{ADM}, \mathrm{pk}_{san}), \sigma_{\mathrm{FULL}}) = 1$*, else if* $\mathsf{SVerify}(\mathrm{pk}_{san}, (1, m, \mathrm{pk}_{san}, \mathrm{pk}_{sig}) = 1$ *then it returns* $\mathtt{San}$*. Note, at least one of these two verifies, because* Judge *is only run on valid pairs* (m, σ).

Theorem 4. *If the underlying signature scheme* S *is UNF-CMA, then the scheme in Construction 1 is unforgeable, immutable, private, accountable and non-interactive publicly accountable.*

Proof. The first four properties are proven in [6], and thus, by Th. 1, Construction 1 is also non-interactive publicly accountable, as Proof returns $\perp$. □

4.2 Construct. 2: Blockwise Non-Interactive Public Accountability

The basic idea of our second construction is depicted in Fig. 5. It satisfies blockwise non-interactive public accountability. Each message first gets a unique random message identifier TAG to prevent mix and match attacks. The signer then signs the fixed blocks together with TAG. Additionally, he binds each of the modifiable blocks separately to the fixed message part and TAG via a blockwise signature. To replace a block, the sanitizer removes the existing blockwise signature and re-signs it using sk_{san}. However, a malicious signer could re-use his tag and include blocks from previous messages which bear a sanitizer's signature. To prevent this, the sanitizer additionally binds $m[i]'$ to a new random tag TAG$'$.

Construction 2 (Blockwise Non-Interact. Publicly Acc. SanSig) *Let* $S = (\mathsf{SKGen}, \mathsf{SSign}, \mathsf{SVerify})$ *be a regular signature scheme. Define the sanitizable signature scheme* $\mathsf{SanSig} = (\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Judge}, \mathsf{Detect})$ *as follows:*

Key Generation: Algorithm KGen_{sig} *generates on input of the security parameter* 1^λ *a key pair* $(\mathrm{pk}_{sig}, \mathrm{sk}_{sig}) \leftarrow \mathsf{SKGen}(1^\lambda)$ *of the underlying signature scheme, and algorithm* KGen_{san} *for input* 1^λ *analogously returns a pair* $(\mathrm{pk}_{san}, \mathrm{sk}_{san}) \leftarrow \mathsf{SKGen}(1^\lambda)$.

Signing: Algorithm Sign *on input* $m \in \{0,1\}^*, \mathrm{sk}_{sig}, \mathrm{pk}_{san}, \text{ADM}$ *draws a random tag* TAG *and computes for all fixed blocks* $m[\text{FIX}]$

$$\sigma_{\text{FIX}} \leftarrow \mathsf{SSign}(\mathrm{sk}_{sig}, (0, m[\text{FIX}], \text{ADM}, \mathrm{pk}_{san}, \text{TAG}))$$

and for each block $m[i]$ *with* $i \in \text{ADM}$*, it computes*

$$\sigma[i] \leftarrow \mathsf{SSign}(\mathrm{sk}_{sig}, (i, m[i], \mathrm{pk}_{san}, \mathrm{pk}_{sig}, \text{TAG}, \perp)).$$

It returns $(m, \sigma) = (m, (\sigma_{\text{FIX}}, (\sigma_i)_{i \in \text{ADM}}, \text{ADM}, \text{TAG}, \perp))$.

Sanitizing: Algorithm Sanit *on input a message* m*, modification instruction* MOD*, a signature* $\sigma = (\sigma_{\text{FIX}}, (\sigma_i)_{i \in \text{ADM}}, \text{ADM}, \text{TAG})$*, keys* pk_{sig} *and* sk_{san} *first verifies that the signature-message pair is valid by running* Verify*. If not, it stops and returns* $\perp$*. It generates* $m' = \text{MOD}(m)$.
If the changed m' *does not conform to* ADM*, it stops and returns* $\perp$*. It then draws a random tag* TAG$'$ *and for each* $(i, m[i]')$ *in* MOD *and computes*

$$\sigma[i]' \leftarrow \mathsf{SSign}(\mathrm{sk}_{san}, (i, m[i]', \mathrm{pk}_{san}, \mathrm{pk}_{sig}, \text{TAG}, \text{TAG}'))$$

For each $(i, m[i]) \in \text{ADM} \setminus \text{MOD}$*, it sets* $\sigma[i]' := \sigma[i]$*. It then outputs* m' *together with* $\sigma' = (\sigma_{\text{FIX}}, (\sigma_i)'_{i \in \text{ADM}}, \text{ADM}, \text{TAG}, \text{TAG}')$.

Verification: Algorithm Verify *on input a message* $m \in \{0,1\}^*$*, a signature* $\sigma = (\sigma_{\text{FIX}}, (\sigma_i)_{i \in \text{ADM}}, \text{ADM}, \text{TAG}, \text{TAG}')$ *and public keys* pk_{sig}*,* pk_{san} *first checks that* $\mathsf{SVerify}(\mathrm{pk}_{sig}, (0, m[\text{FIX}], \text{ADM}, \mathrm{pk}_{san}, \text{TAG}), \sigma_{\text{FIX}}) = 1$*, hence* σ_{FIX} *is a valid signature. For each block* $m[i]$ *with* $i \in \text{ADM}$ *it checks that at*

least one of the algorithms $\mathsf{SVerify}(\mathrm{pk}_{sig}, (i, m[i], \mathrm{pk}_{san}, \mathrm{pk}_{sig}, \mathrm{TAG}, \perp), \sigma[i])$ *or* $\mathsf{SVerify}(\mathrm{pk}_{san}, (i, m[i], \mathrm{pk}_{san}, \mathrm{pk}_{sig}, \mathrm{TAG}, \mathrm{TAG}'), \sigma[i])$ *returns* 1. *If so, it outputs* 1. *Otherwise it returns* 0, *indicating an invalid signature.*

Detect: The Detect *algorithm on input* $(m, \sigma, \mathrm{pk}_{sig}, \mathrm{pk}_{san}, i)$ *returns* `Sig` *if* $\mathsf{SVerify}(\mathrm{pk}_{sig}, (i, m[i], \mathrm{pk}_{san}, \mathrm{pk}_{sig}, \mathrm{TAG}, \perp), \sigma_i) = 1$ *and* `San` *if* $\mathsf{SVerify}(\mathrm{pk}_{san}, (i, m[i]', \mathrm{pk}_{san}, \mathrm{pk}_{sig}, m[\mathrm{FIX}], \mathrm{TAG}, \mathrm{TAG}'), \sigma_i) = 1$. *Note, at least one of the verifications succeeds, as* Detect *is only queried for valid pairs* (m, σ).

Proof: The Proof *algorithm always returns* $\perp$

Judge: The algorithm Judge *on input* $m, \sigma, \mathrm{pk}_{sig}, \mathrm{pk}_{san}$ *verifies that the signature is valid using* Verify. *If not, it aborts returning* $\perp$. *If the signature is valid,* Judge *returns* `Sig`, *if* Detect *marks* **all** *message blocks as* `Sig`, *otherwise* `San`.

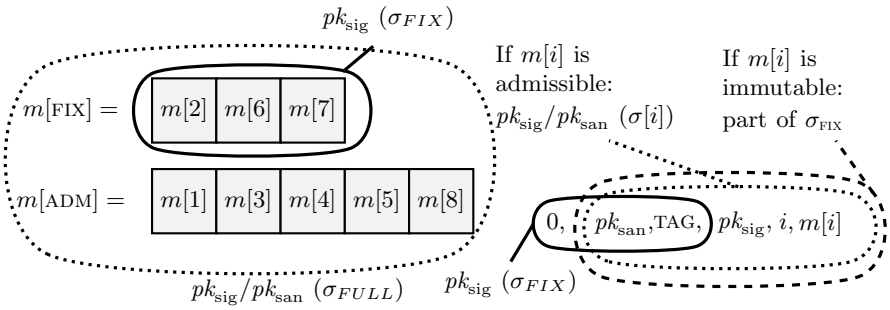

Fig. 4. Construction 1: Blocks 2,6,7 fixed and signed by signer; Complete m signed by either signer or sanitizer.

Fig. 5. Construction 2: Blockwise Public Accountability due to the blockwise signatures $\sigma[i]$ (TAG' not shown)

Theorem 5. *If the underlying signature scheme* S *is UNF-CMA, then the scheme in Construction 2 is unforgeable, immutable, private, accountable and blockwise non-interactive publicly accountable.*

Proof. Blockwise non-interactive public accountability implies non-interactive public accountability and accountability; the latter implies unforgeability [5]. It is thus sufficient to prove privacy, immutability and blockwise non-interactive public accountability. Privacy holds information-theoretically, as the original content of the modified message blocks is not used by the algorithm. The proof for immutability can be done analogously as for Construction 1, only the additional TAG needs to be taken into account. However, TAG does not affect the analysis. It remains to prove blockwise non-interactive public accountability.

Assume, there is an efficient adversary $\mathcal{A}$ against blockwise non-interactive public accountability. Then, we construct an efficient adversary $\mathcal{B}$ against the UNF-CMA of the underlying signature scheme as follows. The adversary $\mathcal{B}$ gets as

input a public key pk and flips a coin b. If $b = 0$, it sets $pk_{\text{sig}} := pk$ and runs SKGen to generate $(pk_{\text{san}}, sk_{\text{san}})$. If $b = 1$, it sets $pk_{\text{san}} := pk$ and runs SKGen to generate $(pk_{\text{sig}}, sk_{\text{sig}})$. Subsequently, to simulate the oracles for $\mathcal{A}$, the algorithm $\mathcal{B}$ runs the algorithms Sign and Sanit according to the specification with the exception that whenever a signature is generated under the secret key sk corresponding to pk, the adversary $\mathcal{B}$ does not generate the signature itself, but instead, it queries its signing oracle. In the end, the adversary $\mathcal{A}$ outputs a triple (pk^*, m^*, σ^*), from which $\mathcal{B}$ extract a valid signature as follows – we distinguish between two cases, a malicious sanitizer attack and a malicious signer attack. The probability that the simulation was done for the wrong case, is exactly $\frac{1}{2}$.

Malicious Sanitizer.

I) The tag TAG^* in (m^*, σ^*, pk^*) has never been returned by the signing oracle. $\mathcal{B}$ returns $(0, m[\text{FIX}]^*, \text{ADM}^*, pk^*_{\text{san}}, \text{TAG}^*, \perp)$ together with σ^*_{FIX} as its forgery.

II) The tag TAG^* used in (m^*, σ^*, pk^*) has been returned by the signing oracle in the ith query. If $pk^*_{\text{san}} \neq pk_{\text{san},i}$ then $(0, m[\text{FIX}]^*, \text{ADM}^*, pk^*_{\text{san}}, \text{TAG}^*, \perp)$ together with σ^*_{FIX} is output. Else, there is a q with $m_i[q] \neq m^*[q]$, but the local signature $\sigma^*[q]$ verifies under the signers public key pk_{sig}. The adversary $\mathcal{B}$ returns $\sigma^*[q]$ together with $(q, m^*[q], pk^*_{\text{san}}, pk_{\text{sig}}, \text{TAG}^*, \perp)$.

Malicious Signer.

I) The tag TAG'^* used in (m^*, σ^*, pk^*) has never been returned by the sanitizing oracle. Then, $\mathcal{B}$ searches for a q where the local signature $\sigma^*[q]$ verifies under the sanitizer's public key pk_{san} and returns $\sigma^*[q]$ together with $(q, m^*[q], pk^*_{\text{san}}, pk_{\text{sig}}, m[\text{FIX}], \text{TAG}^*, \text{TAG}'^*)$.

II) The tag TAG'^* used in (m^*, σ^*, pk^*) has been returned by the sanitizing oracle in the jth query. If $pk^*_{\text{sig}} \neq pk_{\text{sig},j}$ then $(q, m^*[q], pk^*_{\text{san}}, pk_{\text{sig}}, m[\text{FIX}], \text{TAG}^*, \text{TAG}'^*)$ together with $\sigma^*[q]$ is output, for a q, where $\sigma^*[q]$ is a valid signature under pk_{san}. Else if $pk^*_{\text{sig}} = pk_{\text{sig},j}$, then there is a q with $m'_j[q] \neq m^*[q]$, but the local signature $\sigma^*[q]$ verifies under the sanitizer's public key pk_{sig}. The adversary $\mathcal{B}$ returns $\sigma^*[q]$ together with $(q, m^*[q], pk^*_{\text{san}}, pk_{\text{sig}}, \text{TAG}^*)$.

Analysis. We omit the loss in success probability that might occur via TAG collisions, as $\text{TAG} \in \{0,1\}^\lambda$. The adversary $\mathcal{B}$ has then a success probability which is $\frac{1}{2}$ the success probability of $\mathcal{A}$. If $\mathcal{B}$ guesses correctly whether it should embed the key for a malicious sanitizer or a malicious signer attack, then $\mathcal{B}$ is successful, whenever $\mathcal{A}$ is, because the messages output by $\mathcal{B}$ were not queried by $\mathcal{B}$ to its signing oracle before. In detail, if the tag has never been returned by the oracle, then the message is clearly fresh. Else, if the tag has been output to $\mathcal{A}$, then it has not been signed together with the corresponding block. Hence, the message is fresh w.r.t. the block and thus also a successful forgery. □

	KeyGen	Sign			Sanit (10% of $\lvert\text{ADM}\rvert$)			Verify			Detect		
$\lvert\text{ADM}\rvert$ \ ℓ	10/50/100	10	50	100	10	50	100	10	50	100	10	50	100
10% of ℓ	18,649	200	187	200	101	102	101	9	10	13	11	11	13
50% of ℓ	18,649	187	165	187	104	101	111	12	9	10	9	11	12
90% of ℓ	18,649	165	200	165	102	99	98	8	10	14	11	12	16

Table 1. Median Runtime Scheme 1; All in ms

	KeyGen	Sign			Sanit (10% of $\lvert\text{ADM}\rvert$)			Verify			Detect		
$\lvert\text{ADM}\rvert$ \ ℓ	10/50/100	10	50	100	10	50	100	10	50	100	10	50	100
10% of ℓ	18,649	565	2,301	4,569	110	462	932	24	68	126	38	136	301
50% of ℓ	18,649	506	2,127	4,157	454	2,291	5,029	29	86	182	55	222	340
90% of ℓ	18,649	547	2,069	4,164	443	2,106	4,266	27	84	187	53	189	385

Table 2. Median Runtime Scheme 2; All in ms

5 Performance Measurements

We implemented both schemes to demonstrate their practical usability. All tests were performed on an Intel T8300 Dual Core @`2.40` GHz and `4` GiB of RAM, running *Ubuntu* Version 10.04 LTS (64 Bit) and Java version `1.6.0_26-b03`. We utilized a single thread to calculate the signatures; an improvement would be to parallelize signature calculations. The source code is available upon request. We took the median of `100` runs. The signature is an RSA-PSS with a key size of `4096` Bit. Other signatures can be used as well to save space, e.g., aggregate signature schemes [3]. For the tests, we applied our algorithms to messages with 10, 50 and 100 blocks. For any block count, we varied the amount of admissible blocks from 10% to a maximum of 90%, and we sanitized always 10% of the admissible blocks. The results in Tab. 1 and Tab. 2 show that our schemes keep reasonably performant with a high security parameter size and high block counts.

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. `eprint.iacr.org/`.
2. G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik. Sanitizable Signatures. In *ESORICS*, pages 159–177, 2005.
3. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *EUROCRYPT*, pages 416–432, 2003.
4. 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. In *ACNS*, pages 87–104. Springer, 2010.
5. 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.
6. 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.
7. C. Brzuska, M. Fischlin, A. Lehmann, and D. Schröder. Unlinkability of sanitizable signatures. In *PKC*, pages 444–461, 2010.
8. S. Canard, A. Jambert, and R. Lescuyer. Sanitizable signatures with several signers and sanitizers. In *AFRICACRYPT*, pages 35–52, 2012.
9. S. Canard, F. Laguillaumie, and M. Milhau. Trapdoor sanitizable signatures and their application to content protection. In *ACNS*, pages 258–276, 2008.
10. R. M. Caplan. HIPAA. health insurance portability and accountability act of 1996. *Dent Assist*, 72(2):6–8, 1997.
11. E.-C. Chang, C. L Lim, and J. Xu. Short Redactable Signatures Using Random Trees. CT-RSA '09, pages 133–147, Berlin, Heidelberg, 2009. Springer-Verlag.
12. EC. Directive 1999/93/EC from 13 December 1999 on a Community framework for electronic signatures. *Official Journal of the EC*, L 12:12–20, 2000.
13. T. Izu, N. Kunihiro, K. Ohta, M. Sano, and M. Takenaka. Isa. chapter Sanitizable and Deletable Signature, pages 130–144. Springer-Verlag, Berlin, Heidelberg, 2009.
14. R. Johnson, D. Molnar, D. Song, and D.Wagner. Homomorphic signature schemes. In *CT-RSA*, pages 244–262. Springer, Feb. 2002.
15. A. Kundu and E. Bertino. Structural Signatures for Tree Data Structures. In *Proc. of PVLDB 2008*, New Zealand, 2008. ACM.
16. C.M. Laborde. *Electronic Signatures in International Contracts*, volume 4982. Peter Lang, 2010.
17. 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.
18. K. Miyazaki, S. Susaki, M. Iwamura, T. Matsumoto, R. Sasaki, and H. Yoshiura. Digital documents sanitizing problem. Technical report, IEICE, 2003.
19. H. C. Pöhls and F. Höhne. The role of data integrity in EU Digital Signature legislation - achieving Statutory Trust for Sanitizable Signature Schemes. In *Proc. of STM 2011*, LNCS. Springer, 2012.
20. 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.
21. 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, 2012.
22. R. Steinfeld and L. Bull. Content extraction signatures. In *ICISC*, volume 2288, pages 163–205. Springer, 2002.
23. Z.-Y. Wu, C.-W. Hsueh, C.-Y. Tsai, F. Lai, H.-C. Lee, and Y. Chung. Redactable Signatures for Signed CDA Documents. *J. of Med. Systems*, pages 1795–1808, 2012.