This is a preprint. Final version appears in ARES '17 (August 29-September 01, 2017)
DOI: 10.1145/3098954.3104058, ISBN 978-1-4503-5257-4/17/08

Position Paper: The Past, Present, and Future of Sanitizable and Redactable Signatures

Arne Bilzhause*, Henrich C. Pöhls* and Kai Samelin§
*Chair of IT-Security and Institute of IT-Security & Security Law (ISL), University of Passau, Passau, Germany
§IBM Research – Zurich, Rüschlikon, Switzerland and TU Darmstadt, Darmstadt, Germany
Email: {ab,hp@sec.uni-passau.de}, ksa@zurich.ibm.com

***Abstract*—Sanitizable signature schemes (SSS), as well as redactable signature schemes (RSS), gained a lot of attention in the recent past. In a nutshell, both types of signature schemes allow to alter signed data in a controlled way by a, potentially semi-trusted, third party. The resulting signatures still verify. Thus, the authenticity of the subsequently modified content is preserved. In this position paper, we discuss the state-of-the-art, and highlight potential future research opportunities. We hope this work gives rise to additional ideas, real-life use-cases, and interesting upcoming research, helping to bring both primitives into practice. Hence, this paper is meant as a starting point for readers interested in these primitives, looking for new research and application opportunities. In other words, we think that both primitives deserve further attention.**

I. Introduction

Traditional digital signatures become invalid as soon as a single bit of the protected message is changed [71]. However, there is a plethora of real-life use-cases where the alteration of a signed message by an external party is of paramount importance. In the best case, this is possible even without additional interaction with the original signer. Examples of such use-cases include privacy-preserving handling of patient-data, secure routing, work-flows, privacy-preserving document disclosure, privacy credentials, social networks, web-services, and blank signatures [5], [15], [33], [34], [35], [37], [42], [54], [55], [77], [107], [108], [115], [122], [124].

Sanitizable signature schemes (SSS) and redactable signature schemes (RSS) are primitives allowing for such controlled, and non-interactive, alterations. In a nutshell, RSSs allow to remove, i.e., *redact*, parts of a signed message, while SSSs allow a designated third party, named the sanitizer, to change, i.e., *sanitize*, signer-chosen parts of a signed message to different bitstrings. However, even though RSSs and SSSs have a lot of convincing application scenarios, they have not been deployed in practice yet. This is a pity, as in our opinion, both primitives have gained a lot of interest in the recent past from the scientific community, while existing prototype implementations show that both primitives are sufficiently efficient, even from a purely practical point of view. Thus, they can be considered "mature", even though not all opportunities are exhausted yet.

A. Contribution

In this position paper, we review the state-of-the-art of RSSs and SSSs. We also compare both primitives to other related primitives. We point out additional real-life use-cases, but also where they can (and potentially have to) be extended to allow for more application scenarios. This may help to find additional research opportunities, and to increase the visibility of these, in certain contexts very useful, primitives. This may be a stepping stone to bring them into practice. Summarized, this paper is meant as an *easy-to-read starting point* into this interesting topic, proposing additional real-life use-cases, and new research opportunities.

For example, we ask if it is possible to combine RSSs and SSSs, if it possible to have an SSS which is unlinkable and (strongly) invisible simultaneously, or how to construct multi-redacter/multi-signer accountable RSSs, and if SSSs may help to tackle the DNS-enumeration problem [70].

B. Related Primitives

As already pointed out, both RSSs and SSSs, allow to partially delegate signing-rights to other parties. This section is devoted to give an overview over different, yet related, primitives. We only sketch their possibilities, as they have a slightly different focus. In particular, SSSs and RSSs are explicitly designed to alter, i.e., edit, signed data, while the primitives presented now are more meant to "derive" or create new signatures. The current state-of-the-art of RSSs and SSSs (as editable signatures) is discussed in their own sections, i.e., Section III and Section IV.

Due to space constraints, we can only give a very short overview over the related primitives. However, there are already comprehensive overviews, which cover the mentioned primitives in much greater detail [1], [30], [53], [67], [68].

1) Proxy Signature Schemes: Proxy signatures [24], [100] allow to "delegate" signing rights to another party. The second party can then sign messages on behalf of the first entity. However, in the current definition, proxy signatures require that the delegated signing key is generated using an interactive algorithm, while the message space from which the delegatee is allowed to sign messages from is fixed after that setup.

2) Policy-Based Signatures: An additional related primitive are policy-based signature schemes [17]. Here, a trusted authority issues the signing key for each participant, which can only issue signatures for the message space corresponding to the policy encoded into the signing key, while the signatures do not reveal the policy itself.

3) Functional Signatures: They are also signatures where a master signing key can be used to derive restricted so called secondary signing keys [28]. These secondary keys are parametrized by a function f and such a key can only sign

messages in the range of f, i.e., given any m, a key k for function f only allows to produce signatures for $f(m)$.

4) Blank Digital Signature Schemes: In blank digital signature schemes [55], [77], [123], an originator can sign a template which allow the proxy (the party receiving the secret key) to sign a message based on this template. A template can be best interpreted as a document with single/multiple-choice fields.

5) Attribute-Based Signature Schemes: Attribute-based signatures [99] allow users to sign messages which are bound to specific attributes. This type of signature scheme requires that the signer holds the corresponding attributes. More precisely, a signature verification only reveals that the signer has a signing key which fulfills the predicate on the claimed attributes.

6) Transitive Signature Schemes: Assume we have a graph $G = (V, E)$, and each edge $e_{i,j} \in E$ is signed. A transitive signature scheme [20], [102], [125] allows to derive a signature $\sigma_{i,k}$, if there is a path from $v_i \in V$ to $v_k \in V$, even though only signatures for the edges are known. In other words, one can derive signatures on the transitive closure of the graph G.

7) Homomorphic Signature Schemes: Homomorphic signature schemes [26], [27], [74], [84] also allow to derive new signatures from already existing ones. Examples include signing the mean value of grades, or the standard deviation. Related are $\mathcal{P}$-homomorphic signatures [1], where the message derived needs to follow some predicate $\mathcal{P}$, and can be seen as a generalization of RSSs (See Section IV), even though they allow for way more possibilities. Stronger security (in particular privacy) definitions, and impossibility results, have also been presented [7], [8], [52].

8) Additional Related Signature Schemes: There are a lot more related primitives, such as group signature schemes [19], [21], [47], [59], append-only signature schemes [86], threshold signature schemes [60] (and specializations thereof [14], [25], [38]), traceable signature schemes [85], direct anonymous attestation [29], blind signature schemes [46], and generalizations of some of the ideas mentioned [9], [45]. We stress that this list is not exhaustive, but we think that this list provides a good overview, especially as a starting point for the interested reader.

II. Notation

The main security parameter is denoted by $\lambda \in \mathbb{N}$. All algorithms implicitly take 1^λ as an additional input. We write $a \leftarrow A(x)$ if a is assigned to the output of algorithm A with input x. An algorithm is efficient if it runs in probabilistic polynomial time (ppt) in the length of its input. For the remainder of this paper, all algorithms are ppt if not explicitly mentioned otherwise. Most algorithms may return a special error symbol $\perp \notin \{0,1\}^*$, denoting an exception. If S is a set, we write $a \leftarrow S$ to denote that a is chosen uniformly at random from S. For a message $m = (m[1], m[2], \ldots, m[\ell])$, we call $m[i]$ a block, while $\ell \in \mathbb{N}$ denotes the number of blocks in a message m. For a list we require that we have a unique, injective, and efficiently reversible, encoding, which maps the list to $\{0,1\}^*$.

III. Sanitizable Signature Schemes

In this section, we discuss sanitizable signatures schemes (SSS). SSSs have been introduced by Ateniese et al. [5]. When signing, the signer determines which blocks $m[i]$ of the message $m = (m[1], m[2], \ldots, m[i], \ldots, m[\ell])$ can be modified (i.e., are *admissible*). Any such admissible block can be altered to a new bitstring $m[i]' \in \{0,1\}^*$, where $i \in \{1, 2, \ldots, \ell\}$, by a semi-trusted *sanitizer*, also chosen by the signer. This sanitizer holds his own public key. The sanitization process requires the corresponding private key, but does not require the signer's involvement. Sanitization of a message m then results in an altered message $m' = (m[1]', m[2]', \ldots, m[i]', \ldots, m[\ell]')$, where $m[i] = m[i]'$ for every non-admissible block, and also a signature σ', which verifies under the given public keys. Hence, authenticity of the message m' is still ensured. An example workflow is visualized in Figure 1, and Figure 2.

A. The Framework for Sanitizable Signature Schemes

Subsequently, we introduce the framework for SSSs. The definitions are essentially the ones given by Camenisch et al. [36], which are itself based on existing work [31]. However, we need to set some additional notation beforehand. The variable adm contains the set of indices of the modifiable blocks, as well as ℓ denoting the total number of blocks in the message m. For example, let $adm = (\{1, 2, 4\}, 4)$. Then, m must contain four blocks, and all but the third are admissible. The variable MOD is a set containing pairs $(i, m[i]')$ for those blocks that are modified, meaning that $m[i]$ is replaced by $m[i]'$.

Definition 1 (Sanitizable Signatures): A sanitizable signature scheme SSS consists of the ppt algorithms $(\mathsf{SSSPGen}, \mathsf{KGen_{sig}}, \mathsf{KGen_{san}}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$ such that:

$\mathsf{SSSPGen}$. The algorithm $\mathsf{SSSPGen}$, on input security parameter λ, generates the public parameters:

$$\mathsf{pp_{sss}} \leftarrow \mathsf{SSSPGen}(1^\lambda)$$

We assume that $\mathsf{pp_{sss}}$ is implicitly input to all other algorithms.

$\mathsf{KGen_{sig}}$. The algorithm $\mathsf{KGen_{sig}}$ takes the public parameters $\mathsf{pp_{sss}}$, and returns the signer's private key and the corresponding public key:

$$(\mathsf{sk_{sig}}, \mathsf{pk_{sig}}) \leftarrow \mathsf{KGen_{sig}}(\mathsf{pp_{sss}})$$

$\mathsf{KGen_{san}}$. The algorithm $\mathsf{KGen_{san}}$ takes the public parameters $\mathsf{pp_{sss}}$, and returns the sanitizer's private key as well as the corresponding public key:

$$(\mathsf{sk_{san}}, \mathsf{pk_{san}}) \leftarrow \mathsf{KGen_{san}}(\mathsf{pp_{sss}})$$

Sign. The algorithm Sign takes as input a message m, $\mathsf{sk_{sig}}$, $\mathsf{pk_{san}}$, as well as a description adm of the admissible blocks. If $adm(m) = \texttt{false}$, this algorithm returns $\perp$. It outputs a signature:

$$\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk_{sig}}, \mathsf{pk_{san}}, adm)$$

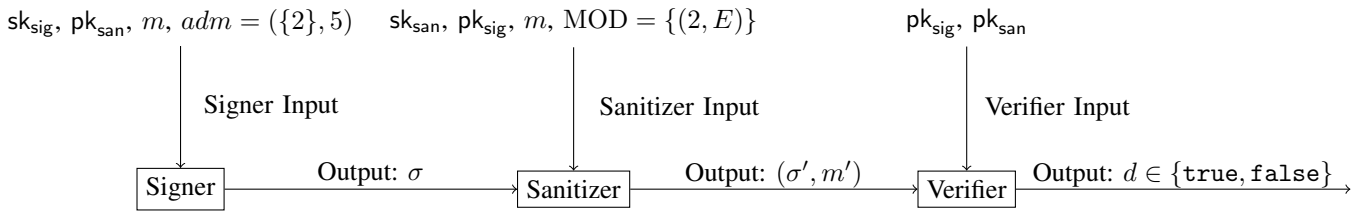

Fig. 1. Example workflow of sanitizable signatures. The message m is set to (H, A, L, L, O). After sanitizing, m' is (H, E, L, L, O).

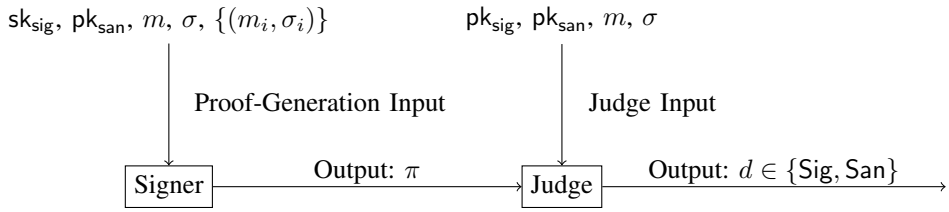

Fig. 2. Proof-generation and Judge.

Sanit. The algorithm Sanit takes a message m, modification instruction MOD, a signature σ, $\mathsf{pk}_{\mathsf{sig}}$, and $\mathsf{sk}_{\mathsf{san}}$. It outputs $m' \leftarrow \mathrm{MOD}(m)$ together with σ':

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

Here, $m' \leftarrow \mathrm{MOD}(m)$ means that the message m is modified to m' according to the modification instruction MOD.

Verify. The algorithm Verify takes as input the signature σ for a message m w.r.t. the public keys $\mathsf{pk}_{\mathsf{sig}}$, and $\mathsf{pk}_{\mathsf{san}}$. It outputs a decision $d \in \{\mathtt{true}, \mathtt{false}\}$:

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

Proof. The algorithm Proof takes as input $\mathsf{sk}_{\mathsf{sig}}$, a message m, a signature σ, a set of polynomially many additional message/signature pairs $\{(m_i, \sigma_i)\}$, and $\mathsf{pk}_{\mathsf{san}}$. It outputs a string $\pi \in \{0,1\}^*$ which can be used by the Judge to decide which party is accountable given a message/signature pair (m, σ):

$$\pi \leftarrow \mathsf{Proof}(\mathsf{sk}_{\mathsf{sig}}, m, \sigma, \{(m_i, \sigma_i) \mid i \in \mathbb{N}\}, \mathsf{pk}_{\mathsf{san}})$$

Judge. The algorithm Judge takes as input a message m, a signature σ, $\mathsf{pk}_{\mathsf{sig}}$, $\mathsf{pk}_{\mathsf{san}}$, as well as a proof π. Note, this means that once a proof π is generated, the accountable party can be derived by anyone for that message/signature pair (m, σ). It outputs a decision $d \in \{\mathsf{Sig}, \mathsf{San}\}$, indicating whether the message/signature pair has been created by the signer, or the sanitizer:

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

B. The Past

The idea of sanitizable signature schemes was introduced by Ateniese et al. [5]. This basic scheme is based upon standard digital signature schemes and chameleon-hashes[1] [88]. In a nutshell, each admissible block is hashed using such a chameleon-hash, while each non-admissible block is hashed using a standard cryptographic hash. Then, all hashes are concatenated and signed using the standard digital signature scheme. The main idea is now, that the sanitizer holds the secret key to the public hashing key and thus can find collisions in the chameleon-hash, which directly corresponds to sanitizing. We stress that the original construction is not secure, as it is neither unforgeable nor accountable in the security model by Brzuska et al. [31], which is in wide use.

1) Standard Security Properties: The first approach to formalize the ideas by Ateniese et al. [5] was given by Brzuska et al. [31]. In particular, they formalized the following properties, which we only sketch informally as it is sufficient for the purpose of this paper.

- *Unforgeability.* If one does not hold any secret keys, one should not be able to come up with a verifying signature σ^* for a message m^* which has neither been authorized by the signer nor the sanitizer.
- *Privacy.* An outsider not holding any private keys should not be able to derive any information about sanitized parts of a message.
- *Transparency.* Transparency is even stronger than privacy. Namely, an outsider not holding any private keys should not even be able to decide whether a signature has been created using the original signing algorithm or through the sanitization algorithm.
- *Immutability.* Immutability requires that only admissible parts of a message can be altered by the sanitizer.
- *Accountability.* If it comes to a dispute, the signer must be able to generate a proof which points to the accountable party. Accountability requires that neither the signer nor the sanitizer can generate "false" proofs pointing to the wrong party for a adversarially generated message/signature pair (m^*, σ^*).

C. The Present

1) Additional Security Properties: The above security properties were later extended to cover more use-cases. In

[1]Chameleon-hashes are a special type of collision-resistant hashes, where hashes are computed using a public key and knowledge of the corresponding secret key allows to find arbitrary collisions

particular, the initial properties were extended for:

- *Unlinkability.* Unlinkability, as a very strong privacy notion, has been introduced by Brzuska et al. [33]. This property guarantees that a sanitized *signature* does not leak from which signature it was created. Thus, even if the original signature is known, it becomes hard to decide if two signatures are linked together (hence the name). Constructions (and stronger definitions) have gained recent attention [35], [63], [95].
- *Non-Interactive Public-Accountability.* From a privacy perspective, transparency seems to be a necessity. However, it turned out that certain use-cases, e.g., for legal compliance, transparency is actually counterproductive [79]. In particular, it may be necessary that it is visible if a signature was sanitized or not. Non-interactive public-accountability is exactly the property achieving this, and was introduced by Brzuska et al. [34], with a construction derived from [32]. Clearly, this property is mutually exclusive to transparency. It can also be facilitated to enforce a multiple-eyes principle [23].
- *Invisibility.* In the original paper, Ateniese et al. [5] introduced the notion of "strong transparency". This notion guarantees that an outsider not holding any private keys cannot derive which blocks are admissible. However, as shown by Pöhls et al. [114], (standard) transparency is not related to this property. Then, Camenisch et al. [36] formalized this idea, and coined it "invisibility", which has, quite recently, been strengthened by Beck et al. [13].
- *Contingency.* In 2009 the "dual of integrity" [116] termed *contingency* [16] has been introduced. Duality means here that something can be proven to be in only one of two states, in this case data either has "integrity" or it is "contingent". Hence, contingency describes the verifiable state that the data's integrity is intended to be unknown, which can be used to circumvent some legal challenges [116], [16], [107]. Pöhls proposed in [107] to technically build this using an SSS.
- *Group-Level Definitions.* All the above definitions are either defined on a per message/signature pair basis, or for each block [34]. However, to improve efficiency, Pöhls et al. proposed to group blocks [49], which also extends to some of the security properties mentioned earlier.
- *Stronger Security Definitions.* The definitions by Brzuska et al. [31] only take the message and signature account. However, as shown by Gong et al. [72], their security model still allows to "fake" adm[2]. Gong et al. thus argue that protecting adm may be necessary, and present a new security framework [72] achieving this. Recently, Krenn et al. [89] further strengthened the security model by also taking the signature into account, much like the difference between unforgeable [71] and strongly unforgeable standard signatures [4], [80]. Then, de Meer et al. showed how to transform SSSs into RSSs, if certain privacy definitions are altered in a non-standard way [50].

2) Multiple Signers and Sanitizers: As already clear from the given framework, SSSs are normally defined in a way that only *one* signer and *one* sanitizer are considered. This may not be enough for certain use-cases. Thus, the idea of *trapdoor* SSSs has been introduced by Canard et al. [41]. In this primitive, the signer can, after signature generation, grant additional sanitizers the possibility to sanitize a signature. Later, this primitive has been extended to allow for more efficient instantiations, and accountability [32], [94], [126]. This has later been extended to also cover multiple signers and unlinkability [40].

3) Limiting the Sanitizer: In the above schemes, the sanitizer is allowed to alter the admissible blocks to arbitrary bitstrings. Clearly, this is not always wanted. To tackle this situation, new schemes have been proposed which allow to limit the sanitizer to signer-chosen values for certain blocks [39], [58], [87].

4) Sanitization of Encrypted Data: Recently, the idea of sanitizing encrypted data was introduced [11], [48], [62], [65] and also a patent in this area exists [66]. In this primitive, the sanitizer does not learn which data is sanitized, i.e., they add an additional layer of confidentiality.

5) Implementations: To prove that SSSs are practical, they were implemented on a wide variety of devices, ranging from smart cards [110] to potent desktop PCs [13], [49], even in very different contexts such as XML-signatures [114]. All implementations presented so far show that SSSs can be considered efficient.

D. The Future

1) Legal Implications and Standardization: On the one hand, from a purely practical perspective, it is necessary to know which legal consequences arise, if SSSs are deployed in real-world application. As it turned out, non-interactive public-accountability seems to be necessary [109], [79], [**?**] to achieve a level of trust which meets legal requirements, as (qualified) digital signatures are equivalent to handwritten ones in court. However, as SSSs have not yet been deployed, there is still a lot of discussion missing here.

On the other hand, there are a lot of competing security definitions with subtleties that are of importance, as this section has already shown. We think that standardizing SSSs may help here to establish a minimum baseline for a common understanding especially for legal arguments.

2) New Application Scenarios:

- *Receipts.* Assume that you buy two items at a store, e.g., a TV and a toaster. However, at home it turns out that the wrong TV was bought. Normally, one returns to the store, and exchanges the TV for a different one. Then, however, the receipt needs to be changed to prevent fraud. SSSs can help here: if an item is exchanged, the receipt can simply be altered for the new item, without the need to give the customer two different receipts.
  This kind of idea may be useful in other contexts as well, e.g., supply-chain management.
- *Data Loss Prevention.* The main challenges for data loss prevention are clearly confidentiality and availability of data. However, one form of data loss is the so called *data leakage* that can not only affect *data at rest* (e.g. theft of database entries), but also *data in motion* (e.g. data sent by mail, ftp, or peer-to-peer) [97]. And in many cases the

[2]However, Brzuska et al. explicitly require that adm is always correctly recoverable [31].

integrity and authenticity of sent messages are crucial for the receiving side of communication.
Assume that some data is classified to be for internal use only and hence never should leave your network. The first and most obvious approach would be to never send the data. This would require all parts of a system (e.g. applications and/or users) to be aware of the data classification and the according rules (which might simply not be possible in many systems). A second approach would be to implement a central entity that checks the outgoing communication and alters or removes all classified data. In case of standard signatures, two problems arise for the integrity and authenticity of the sent messages: either the central entity invalidates existing signatures (i.e. by altering messages signed by the original data source), or the central entity has to sign the message again with its private key after a change was performed. In the latter case, the trust relation between receiver and data source cannot be verified anymore (i.e. the altered version of the message was never signed by the original data source). SSSs can help here: by making the central entity an authorized sanitizer, changes on outgoing messages can be performed without invalidating the signature, while the trust relation between receiver (verifier) and original data source (signer) can be preserved. Notice that RSSs could be used in a similar manner, however this will limit the options for the central entity (i.e. only removal of message parts is possible). A related approach [22] was presented for web-services in combination with a proxy server (sanitizer).

- *Preventing DNS-Zone Enumeration.* The DNS-Zone enumeration-problem essentially says that certain DNS-extensions, such as DNSSEC, allow to enumerate DNS-zones [70] by provoking (negative) responses from DNS-servers, eventually allowing to enumerate all zones. As also proven by Goldberg et al. [70], there is no simpler solution than public-key cryptography to this problem. Thus, SSSs may offer a direct solution here: each response is pre-computed for each domain by the corresponding zone's secret key. For non-existing zones, however, the DNS-server queried sanitizes a (pre-computed and signed by the root's DNS-server) message $m = (\text{domain}, \text{“BLANK”}, \text{does not exist}$ to $m' = (\text{domain}, \text{“QUERY”}, \text{does not exist})$, where QUERY is the domain to be resolved, and only $m[2]$ is admissible. To prevent "false" responses by corrupt servers, one may restrict $m[2]$ to be an non-existing domain, which can be set up by the administrator. This can, e.g., be achieved using the "restricting the sanitizer"-extensions discussed earlier.

3) Future Research Opportunities and Questions:

- *Seals.* The current definition of transparency prohibits that a legitimate sanitization makes the verifier able to decide whether that action took place. However, it may also be useful to have an extension which allows that, if certain signer-chosen admissible blocks are sanitized, transparency is lost. This may, e.g., be the case in a break-glass scenario, where certain changes to a signed message are necessary, as extraordinary circumstances force the sanitizer to do this, e.g., if a name of a list is pseudonymized to account for data protection laws. A natural extension is to also hide which changes would break that "seal" – only if that seal is broken, the verifier then learns where the seal was. An even stronger definition would only reveal that the seal was broken, but not where.
- *Invisible and Unlinkable* SSSs. There are already invisible [36] and unlinkable [33] SSSs. However, there are no constructions which achieve both properties simultaneously. Thus, it is clearly a question if both security properties can be combined. It is also an open question, if *strongly* unlinkable [35], yet transparent SSSs can be constructed.
- *Black-Box Accountability.* SSSs can also be used to make RSSs accountable [112]. It is therefore a natural question, if they can be used to make other (non-accountable) primitives accountable as well, maybe even using the same underlying construction idea, and for multiple sanitizers.
- *Levels of Accountability.* For SSS the non-interactive public form of accountability has been introduced [34]. It prohibits the property of transparency that has been widely discussed. Still, it remains an open question if there are useful applications for notions between these two extremes, and if such a notion can be extended to one of a SSS's core property: privacy, i.e., whether there exists an application where an outsider must know whether a block/message was altered (and not the signature, as for non-interactive public-accountability), but cannot derive any additional useful information.
- *Time-Locked Sanitization.* It may also be interesting if it is possible to construct SSSs which allow that only at some later point in time the sanitizer can sanitize. This can even be extended in way such that at some later point in time sanitization is no longer possible.

IV. Redactable Signature Schemes

As already clarified, RSSs allow to remove (and only remove) certain parts of a message m, without interaction with the original signer. This redaction can be performed by any party, i.e., no additional key pair is required. As for SSSs, the resulting signatures still verify. An example workflow for RSSs is given in Figure 3.

A. The Framework of Redactable Signature Schemes

The following definitions for RSSs are compiled from [30], [118], but extended to support parameter generation to match the definitions of SSSs.

*Definition 2 (Redactable Signature Scheme (*RSS*)):* A redactable signature scheme RSS consists of five algorithms, i.e., $(\mathsf{RSSPGen}, \mathsf{KGen}_{\mathsf{sig}}, \mathsf{Sign}, \mathsf{Redact}, \mathsf{Verify})$, such that:

$\mathsf{RSSPGen}$**.** The algorithm RSSPGen generates the public parameters:

$$\mathsf{pp}_{\mathsf{rss}} \leftarrow \mathsf{RSSPGen}(1^\lambda)$$

We assume that $\mathsf{pp}_{\mathsf{rss}}$ is implicitly input to all other algorithms.

$\mathsf{KGen}_{\mathsf{sig}}$**.** The signer generates a key pair, based on the security parameter λ:

$$(\mathsf{pk}_{\mathsf{sig}}, \mathsf{sk}_{\mathsf{sig}}) \leftarrow \mathsf{KGen}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{rss}})$$

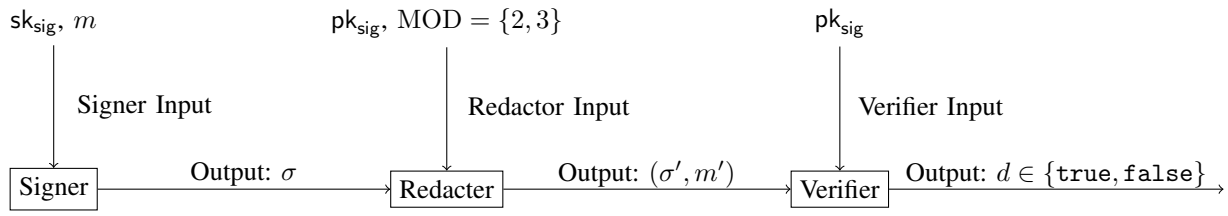

Fig. 3. Example workflow of redactable signatures. The message m is set to $(I, do, not, like, tomatoes)$. After redacting, m' is $(I, \perp, \perp, like, tomatoes)$. The redaction information red is omitted for brevity. If $\perp$ is visible, depends on the scheme, e.g., m' could also be $(I, \perp, like, tomatoes)$ or even $(I, like, tomatoes)$

Sign. The Sign algorithm takes as input a message m, adm, and $\mathsf{sk}_{\mathsf{sig}}$. It outputs a signature σ, and some redaction information red:

$$(\sigma, red) \leftarrow \mathsf{Sign}(m, \mathsf{sk}_{\mathsf{sig}}, adm)$$

Redact. Algorithm Redact takes a message m, MOD, redaction information red, a signature σ, and $\mathsf{pk}_{\mathsf{sig}}$. It modifies the message m according to the modification MOD. Redact outputs $m' \leftarrow \mathrm{MOD}(m)$, red', and σ':

$$(m', \sigma', red') \leftarrow \mathsf{Redact}(m, \mathrm{MOD}, \sigma, red, \mathsf{pk}_{\mathsf{sig}})$$

Verify. Algorithm Verify outputs a decision $d \in \{\texttt{true}, \texttt{false}\}$ verifying the correctness of a signature σ for a message m w.r.t. a public key $\mathsf{pk}_{\mathsf{sig}}$:

$$d \leftarrow \mathsf{Verify}(m, \sigma, \mathsf{pk}_{\mathsf{sig}})$$

B. The Past

The idea of redactable signature schemes was introduced by Steinfeld et al. [121], and, with a slightly different focus, by Johnson et al. [84], even though some prior work on authenticated data-structures was already present [73].

1) Standard Security Properties: The first complete approach to formalize certain security properties was given by Brzuska et al. [30], which, however, focus on trees instead of lists. In particular, Brzuska et al. formalized the following properties, which we only sketch informally as it is sufficient for the purpose of this paper.

- *Unforgeability.* If one does not hold any secret keys, one should only be able to derive signatures for messages which are explicitly derivable by redaction, and no other ones.
- *Privacy.* An outsider not holding any private keys should not be able to derive any information about redacted parts of a message.
- *Transparency.* Transparency is even stronger than privacy. Namely, an outsider not holding any private keys should not even be able to decide whether a signature has been created using the original signing algorithm or through the redaction algorithm. There are also some weaker definitions, which allow transparency for a certain amount of redacted blocks [76].

Clearly, these properties are akin to the definitions for SSSs. However, as RSSs allow for *public* redactions, i.e., there is no other entity holding any secret keys, there is no need to define security against another insider, simplifying the security model significantly.

C. The Present

As for SSSs, RSSs have then been extended for additional possibilities, and security. A generalized framework has later been given by Derler et al. [57], which covers most of the following additional properties from existing constructions, e.g., [43], [83], [10], [81], [82]

1) Additional Security Properties: The above security properties were later extended to cover more use-cases. In particular, the initial properties were extended for:

- *Unlinkability.* As for SSSs, unlinkability of RSSs requires that a signature does not leak from which signature it was derived [37]. This is clearly a very strong privacy notion, and not many constructions achieve this notion [1], [7], [8], especially if their target is slightly different.
- *Updatable and Mergeable Redactable Signatures.* RSSs, as their very name suggests, only allow to remove blocks. This, however, was extended to allow updating signature by adding new elements [96], but also to merge signatures derived from the same origin [111]. This is essentially a mixture between RSSs and append-only signature schemes [86], borrowing ideas from "incremental cryptography" [18].
- *Adding Zero-Knowledge.* Some of the ideas have also been extended into the zero-knowledge realm, allowing for order-queries, closest-neighbor searches, yet also zero-knowledge sets [67], [69], [68], [101], and generalizations of the underlying primitives [44].
- *Disclosure Control.* The first definitions allowed that every block of a message is redactable. However, there are application scenarios where either the signer, or even some other intermediate party, wants to prohibit that certain blocks can be redacted [75], [98], [105], [104], [103], [117].
- *Accountable* RSS. Even though RSSs are very useful primitives, their original definition does not consider accountability. This situation has been tackled by Pöhls and Samelin which show how to use SSSs to make RSSs accountable [112]. A related direction are signer-anonymous RSSs, where a verifier cannot decide which signing key was used [56], but can be traced by a trusted by party.
- *Complex Data-Structures and Generalizations.* The idea of RSSs has also been extended to cover more complex data-structures than sets [106] and lists, such as trees, graphs, and forests [12], [61], [78], [92], [93], [90], [91], [114], [113], [51], and related ideas, e.g., dependencies between blocks [119].

2) Implementations: As for SSSs, a lot of work was put into proving that RSSs are practical. Namely, implementations have been presented for smart cards [110], small sensor-nodes [64], desktop PCs [91] (and some later follow-ups by Kundu et al.) to powerful servers used for hospitals [124]. These implementations show that RSSs are practical.

D. The Future

1) Legal Implications and Standardization: As for SSSs, RSSs clearly help protecting data, i.e., may help to fulfill existing data-protection laws [120]. However, also RSSs have not been standardized yet. Therefore, the same arguments and wishes as for SSSs apply: Standardization would provide a common baseline for practitioners and cryptographers.

2) Social Implications and Human Computer Interaction (HCI): Even more for RSSs than for SSSs their actions are often explained with a physical metaphor of redacting using black ink. Clearly this seems helpful to convey the meaning, however as we have seen digital redaction going wrong a powerful tools of RSS needs to be solidly understood. First research in this direction is under way [3], [2] and the results shall be carefully considered when moving ahead to not loose the link to the human users.

3) New Application Scenarios:

- *Redactable Commercials for Ad-Blocking in Authentic Content.* There is already the concept of redactable blockchains [6]. Following their ideas, it may be reasonable to use RSSs to redact certain commercials from a website at a client, e.g., an employee at a company which disallows certain commercials of questionable nature. This may even be paired with a special redaction key, which is only given to the company, if it pays for it. This would allow to block advertisements by redacting them from authenticity protected content.
- *Content Distribution Networks.* Let S be a set. Assume that this set is replicated across multiple servers, e.g., in the cloud. Further assume that these servers form a CDN. Mergeable RSSs [111] allow a client to request different *authenticated* subsets $S_i \subset S$ from different servers to increase download speed but verify all individual downloads as authentic, and then later merge them before further distributing a larger blob of authenticated data.

4) Future Research Opportunities and Questions:

- *Combining* RSS*s and* SSS*s.* As already argued, SSSs allow to alter (not redact) admissible blocks, while RSSs allow only complete removal of blocks. It is therefore a natural question to ask, if both primitives can be combined, i.e., if complete removal and sanitization are possible at the same time, maybe even with some of the extensions discussed earlier.
- *Multi-Signer/Sanitizer* RSS*s.* The notion of accountable RSSs currently only supports one signer and one sanitizer [112]. We therefore ask, if it possible to combine the ideas given in [40], [56], [112] to allow a scheme which has multiple signers, and multiple sanitizers, in an accountable way.
- *Functional* RSS*s.* Assume that a message m consists of three parts. Further assume that the signer wants that either $m[3]$ (or $m[1]$) can be redacted, while if $m[2]$ is removed, also $m[3]$ must be redacted. This may, e.g., be useful if patient data is redacted. This idea is not new per se [119], but it remains the question whether there exists transparent (or even unlinkable) constructions, and if more expressive and useful "functions" exists.

V. Conclusion

In this position paper, we gave an overview of sanitizable signature schemes and redactable signature schemes. We kept it at an introductory-level, still providing a clear separation between the two concepts and achieving a broad coverage of the current body of literature. We compared those primitives to other related schemes, and discussed their current state-of-the-art. Based on arising application scenarios, we identified additional research opportunities, but also stated what, in our opinion, is missing to bring both primitives into practice.

VI. Acknowledgments

H. C. Pöhls was supported by EU H2020 project PRISMACLOUD: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644962. K. Samelin was supported by the EU ERC PERCY, grant agreement n°32131.

We also thank Daniel Slamanig for some interesting discussions regarding this paper.

References

[1] J. H. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, A. Shelat, and B. Waters. Computing on authenticated data. In *TCC*, pages 1–20, 2012.

[2] A. Alaqra, S. Fischer-Hübner, T. Groß, T. Lorünser, and D. Slamanig. *Signatures for Privacy, Trust and Accountability in the Cloud: Applications and Requirements*. Springer, 2016.

[3] A. Alaqra, S. Fischer-Hübner, J. S. Pettersson, and E. Wästlund. Stakeholders perspectives on malleable signatures in a cloud-based ehealth scenario. In *HAISA*, pages 220–230, 2016.

[4] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *EUROCRYPT*, pages 83–107, 2002.

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

[6] G. Ateniese, B. Magri, D. Venturi, and E. R. Andrade. Redactable blockchain - or - rewriting history in bitcoin and friends. *IACR Cryptology ePrint Archive*, 2016:757, 2016.

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

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

[9] M. Backes, Ö. Dagdelen, M. Fischlin, S. Gajek, S. Meiser, and D. Schröder. Operational signature schemes. *IACR Cryptology ePrint Archive*, 2014:820, 2014.

[10] M. Backes, S. Meiser, and D. Schröder. Delegatable functional signatures. *IACR Cryptology ePrint Archive*, 2013:408, 2013.

[11] C. Badertscher, C. Matt, and U. Maurer. Strengthening access control encryption. *IACR Cryptology ePrint Archive*, 2017:429, 2017.

[12] D. Bauer, D. M. Blough, and A. Mohan. Redactable signatures on data with dependencies and their application to personal health records. In *WPES*, pages 91–100, 2009.

[13] M. T. Beck, J. Camenisch, D. Derler, S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig. Practical strongly invisible and strongly accountable sanitizable signatures. In *ACISP Part I*, pages 437–452, 2017.

[14] M. T. Beck, S. Krenn, F.-S. Preiss, and K. Samelin. Practical signing-right revocation. In *Trust*, pages 21–39, 2016.

[15] A. Becker and M. Jensen. Secure combination of xml signature application with message aggregation in multicast settings. In *ICWS*, pages 531–538, 2013.

[16] M. Bedner and T. Ackermann. Schutzziele der IT-Sicherheit. *DuD*, 34(5):323–328, 2010.

[17] M. Bellare and G. Fuchsbauer. Policy-based signatures. In *PKC*, pages 520–537, 2014.

[18] M. Bellare, O. Goldreich, and S. Goldwasser. Incremental cryptography: The case of hashing and signing. In *CRYPTO*, pages 216–233, 1994.

[19] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT*, pages 614–629, 2003.

[20] M. Bellare and G. Neven. Transitive signatures: new schemes and proofs. *IEEE Trans. Information Theory*, 51(6):2133–2151, 2005.

[21] P. Bichsel, J. Camenisch, G. Neven, N. P. Smart, and B. Warinschi. Get shorty via group signatures without encryption. In *SCN*, pages 381–398, 2010.

[22] A. Bilzhause. Property based policies for malleable signatures - exemplified by a web service proxy use case. Master's thesis, University of Passau, 10 2013.

[23] A. Bilzhause, M. Huber, H. C. Pöhls, and K. Samelin. Cryptographically Enforced Four-Eyes Principle. In *ARES*, pages 760–767, 2016.

[24] A. Boldyreva, A. Palacio, and B. Warinschi. Secure proxy signature schemes for delegation of signing rights. *J. Cryptology*, 25(1):57–115, 2012.

[25] D. Boneh, X. Ding, G. Tsudik, and C.-M. Wong. A method for fast revocation of public key certificates and security capabilities. In *USENIX*, 2001.

[26] D. Boneh and D. M. Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In *PKC*, pages 1–16, 2011.

[27] X. Boyen, X. Fan, and E. Shi. Adaptively secure fully homomorphic signatures based on lattices. *IACR Cryptology ePrint Archive*, 2014:916, 2014.

[28] E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In *PKC*, pages 501–519, 2014.

[29] E. F. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *CCS*, pages 132–145, 2004.

[30] C. Brzuska, H. Busch, Ö. 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, 2010.

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

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

[33] C. Brzuska, M. Fischlin, A. Lehmann, and D. Schröder. Unlinkability of Sanitizable Signatures. In *PKC*, pages 444–461, 2010.

[34] C. Brzuska, H. C. Pöhls, and K. Samelin. Non-Interactive Public Accountability for Sanitizable Signatures. In *EuroPKI*, pages 178–193, 2012.

[35] C. Brzuska, H. C. Pöhls, and K. Samelin. Efficient and Perfectly Unlinkable Sanitizable Signatures without Group Signatures. In *EuroPKI*, pages 12–30, 2013.

[36] J. Camenisch, D. Derler, S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig. Chameleon-hashes with ephemeral trapdoors - and applications to invisible sanitizable signatures. In *PKC, Part II*, pages 152–182, 2017.

[37] J. Camenisch, M. Dubovitskaya, K. Haralambiev, and M. Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In *ASIACRYPT*, pages 262–288, 2015.

[38] J. Camenisch, A. Lehmann, G. Neven, and K. Samelin. Virtual smart cards: How to sign with a password and a server. In *SCN*, pages 353–371, 2016.

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

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

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

[42] S. Canard and R. Lescuyer. Protecting privacy by sanitizing personal data: a new approach to anonymous credentials. In *ASIACCS*, pages 381–392, 2013.

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

[44] M. Chase, A. Healy, A. Lysyanskaya, T. Malkin, and L. Reyzin. Mercurial commitments with applications to zero-knowledge sets. *J. Cryptology*, 26(2):251–279, 2013.

[45] M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn. Malleable signatures: New definitions and delegatable anonymous credentials. In *CSF*, pages 199–213, 2014.

[46] D. Chaum. Blind signatures for untraceable payments. In *Crypto*, pages 199–203, 1982.

[47] D. Chaum and E. van Heyst. Group signatures. In *Eurocrypt*, pages 257–265, 1991.

[48] I. Damgård, H. Haagh, and C. Orlandi. Access control encryption: Enforcing information flow with cryptography. In *TCC-B*, pages 547–576, 2016.

[49] H. de Meer, H. C. Pöhls, J. Posegga, and K. Samelin. Scope of security properties of sanitizable signatures revisited. In *ARES*, pages 188–197, 2013.

[50] H. de Meer, H. C. Pöhls, J. Posegga, and K. Samelin. On the relation between redactable and sanitizable signature schemes. In *ESSoS*, pages 113–130, 2014.

[51] H. de Meer, H. C. Pöhls, J. Posegga, and K. Samelin. Redactable signature schemes for trees with signer-controlled non-leaf-redactions. In *E-Business and Telecommunications*, volume 455 of *CCIS*, pages 155–171. Springer, 2014.

[52] B. Deiseroth, V. Fehr, M. Fischlin, M. Maasz, N. F. Reimers, and R. Stein. Computing on authenticated data for adjustable predicates. In *ACNS*, pages 53–68, 2013.

[53] D. Demirel, D. Derler, C. Hanser, H. C. Pöhls, D. Slamanig, and G. Traverso. PRISMACLOUD D4.4: Overview of Functional and Malleable Signature Schemes. Technical report, H2020 Prismacloud, www.prismacloud.eu, 2015.

[54] D. Derler, C. Hanser, H. C. Pöhls, and D. Slamanig. Towards authenticity and privacy preserving accountable workflows. In *Privacy and Identity Management. Time for a Revolution?*, pages 170–186, 2015.

[55] D. Derler, C. Hanser, and D. Slamanig. Blank digital signatures: Optimization and practical experiences. In *Privacy and Identity Management for the Future Internet in the Age of Globalisation*, pages 201–215. Springer, 2014.

[56] D. Derler, S. Krenn, and D. Slamanig. Signer-anonymous designated-verifier redactable signatures for cloud-based data sharing. In *CANS*, pages 211–227, 2016.

[57] D. Derler, H. C. Pöhls, K. Samelin, and D. Slamanig. A general framework for redactable signatures and new constructions. In *ICISC*, pages 3–19, 2015.

[58] D. Derler and D. Slamanig. Rethinking privacy for extended sanitizable signatures and a black-box construction of strongly private schemes. In *ProvSec*, pages 455–474, 2015.

[59] D. Derler and D. Slamanig. Fully-anonymous short dynamic group signatures without encryption. *IACR Cryptology ePrint Archive*, 2016:154, 2016.

[60] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Crypto*, pages 307–315, 1989.

[61] A. Erwig, M. Fischlin, M. Hald, D. Hehm, R. Kiel, F. Kübler,

M. Kümmerlin, J. Laenge, and F. Rohrbach. Redactable graph hashing, revisited. In *ACISP Part II*, pages 398–405, 2017.

[62] V. Fehr and M. Fischlin. Sanitizable signcryption: Sanitization over encrypted data (full version). Cryptology ePrint Archive, Report 2015/765, 2015. http://eprint.iacr.org/.

[63] N. Fleischhacker, J. Krupp, G. Malavolta, J. Schneider, D. Schröder, and M. Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In *PKC Part I*, pages 301–330, 2016.

[64] C. Frädrich, H. C. Pöhls, W. Popp, N. Rakotondravony, and K. Samelin. Integrity and authenticity protection with selective disclosure control in the cloud & iot. In *ICICS*, pages 197–213, 2016.

[65] G. Fuchsbauer, R. Gay, L. Kowalczyk, and C. Orlandi. Access control encryption for equality, comparison, and more. In *PKC*, pages 88–118, 2017.

[66] S. Gajek, J. Seedorf, and Ö. Dagdelen. Method and system for modifying an authenticated and/or encrypted message. https://www.google.com/patents/EP2719149B1?cl=en [last accessed: Jan. 2017], Sep. 2015. EP Patent 2,719,149.

[67] E. Ghosh, M. T. Goodrich, O. Ohrimenko, and R. Tamassia. Fully-dynamic verifiable zero-knowledge order queries for network data. *ePrint*, 2015:283, 2015.

[68] E. Ghosh, O. Ohrimenko, and R. Tamassia. Verifiable member and order queries on a list in zero-knowledge. *ePrint*, 2014:632, 2014.

[69] E. Ghosh, O. Ohrimenko, and R. Tamassia. Efficient verifiable range and closest point queries in zero-knowledge. *PoPETs*, 2016(4):373–388, 2016.

[70] S. Goldberg, M. Naor, D. Papadopoulos, L. Reyzin, S. Vasant, and A. Ziv. NSEC5: provably preventing DNSSEC zone enumeration. In *NDSS*, 2015.

[71] S. Goldwasser, S. Micali, and R. L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, 17:281–308, 1988.

[72] J. Gong, H. Qian, and Y. Zhou. Fully-secure and practical sanitizable signatures. In *InsCrypt*, volume 6584, pages 300–317, 2011.

[73] M. T. Goodrich, R. Tamassia, and N. Triandopoulos. Efficient authenticated data structures for graph connectivity and geometric search problems. *Algorithmica*, 60(3):505–552, 2011.

[74] S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. In *STOC*, pages 469–477, 2015.

[75] 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.

[76] S. Haber, W. Horne, and M. Zhang. Efficient transparent redactable signatures with a single signature invocation. *IACR Cryptology ePrint Archive*, 2016:1165, 2016.

[77] C. Hanser and D. Slamanig. Blank digital signatures. In *AsiaCCS*, pages 95 – 106. ACM, 2013.

[78] S. Hirose and H. Kuwakado. Redactable signature scheme for tree-structured data based on merkle tree. In *SECRYPT*, pages 313–320, 2013.

[79] F. Höhne, H. C. Pöhls, and K. Samelin. Rechtsfolgen editierbarer signaturen. *DuD*, 36(7):485–491, 2012.

[80] Q. Huang, D. S. Wong, and Y. Zhao. Generic transformation to strongly unforgeable signatures. In *ACNS 2007*, pages 1–17, 2007.

[81] T. Izu, M. Izumi, N. Kunihiro, and K. Ohta. Yet another sanitizable and deletable signatures. In *AINA Workshops*, pages 574–579, 2011.

[82] T. Izu, N. Kunihiro, K. Ohta, M. Sano, and M. Takenaka. Yet another sanitizable signature from bilinear maps. In *Ares*, pages 941–946, 2009.

[83] T. Izu, N. Kunihiro, K. Ohta, M. Takenaka, and T. Yoshioka. A sanitizable signature scheme with aggregation. In *ISPEC*, pages 51–64, 2007.

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

[85] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *Eurocrypt*, pages 571–589, 2004.

[86] E. Kiltz, A. Mityagin, S. Panjwani, and B. Raghavan. Append-only signatures. In *ICALP*, pages 434–445, 2005.

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

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

[89] S. Krenn, K. Samelin, and D. Sommer. Stronger security for sanitizable signatures. In *DPM*, pages 100–117, 2015.

[90] A. Kundu, M. J. Atallah, and E. Bertino. Leakage-free redactable signatures. In *Codaspy*, pages 307–316, 2012.

[91] A. Kundu and E. Bertino. Structural signatures for tree data structures. *PVLDB*, 1(1):138–150, 2008.

[92] A. Kundu and E. Bertino. How to authenticate graphs without leaking. In *EDBT*, pages 609–620, 2010.

[93] A. Kundu and E. Bertino. Privacy-preserving authentication of trees and graphs. *Int. J. Inf. Sec.*, 12(6):467–494, 2013.

[94] J. Lai, X. Ding, and Y. Wu. Accountable trapdoor sanitizable signatures. In *ISPEC*, pages 117–131, 2013.

[95] R. W. F. Lai, T. Zhang, S. S. M. Chow, and D. Schröder. Efficient sanitizable signatures without random oracles. In *ESORICS-1*, pages 363–380, 2016.

[96] S. Lim, E. Lee, and C.-M. Park. A short redactable signature scheme using pairing. *SCN*, 5(5):523–534, 2012.

[97] S. Liu and R. Kuhn. Data loss prevention. *IT professional*, 12(2), 2010.

[98] J. Ma, J. Liu, M. Wang, and W. Wu. An efficient and secure design of redactable signature scheme with redaction condition control. In *GPC*, pages 38–52, 2017.

[99] H. K. Maji, M. Prabhakaran, and M. Rosulek. Attribute-based signatures. In *CT-RSA*, pages 376–392, 2011.

[100] M. Mambo, K. Usuda, and E. Okamoto. Proxy signatures for delegating signing operation. In *CCS*, pages 48–57, 1996.

[101] S. Micali, M. O. Rabin, and J. Kilian. Zero-knowledge sets. In *FOCS*, pages 80–91, 2003.

[102] S. Micali and R. L. Rivest. Transitive signature schemes. In *CT-RSA*, pages 236–243, 2002.

[103] K. Miyazaki, G. Hanaoka, and H. Imai. Digitally signed document sanitizing scheme based on bilinear maps. In *AsiaCCS*, pages 343–354, 2006.

[104] K. Miyazaki, G. Hanaoka, and H. Imai. Invisibly sanitizable digital signature scheme. *IEICE Transactions*, 91-A(1):392–402, 2008.

[105] 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.

[106] R. Nojima, J. Tamura, Y. Kadobayashi, and H. Kikuchi. A storage efficient redactable signature in the standard model. In *ISC*, pages 326–337, 2009.

[107] H. C. Pöhls. Contingency revisited: Secure construction and legal implications of verifiably weak integrity. In *IFIPTM*, pages 136–150, 2013.

[108] H. C. Pöhls, A. Bilzhause, K. Samelin, and J. Posegga. Sanitizable signed privacy preferences for social networks. In *DICCDI*, LNI, 2011.

[109] H. C. Pöhls and F. Höhne. The role of data integrity in EU digital signature legislation - achieving statutory trust for sanitizable signature schemes. In *STM*, pages 175–192, 2011.

[110] H. C. Pöhls, S. Peters, K. Samelin, J. Posegga, and H. de Meer. Malleable signatures for resource constrained platforms. In *WISTP*, pages 18–33, 2013.

[111] H. C. Pöhls and K. Samelin. On updatable redactable signatures. In *ACNS*, pages 457–475, 2014.

[112] H. C. Pöhls and K. Samelin. Accountable redactable signatures. In *ARES*, pages 60–69, 2015.

[113] H. C. Pöhls, K. Samelin, H. de Meer, and J. Posegga. Flexible

redactable signature schemes for trees - extended security model and construction. In *SECRYPT*, pages 113–125, 2012.

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

[115] S. Rass and D. Slamanig. *Cryptography for Security and Privacy in Cloud Computing*. Artech House, 2013.

[116] M. Rost and A. Pfitzmann. Datenschutz-schutzziele — revisited. *DuD*, 33(6):353–358, 2009.

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

[118] 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.

[119] D. Slamanig and S. Rass. Generalizations and extensions of redactable signatures with applications to electronic healthcare. In *CMS*, pages 201–213, 2010.

[120] D. Slamanig and S. Rass. Redigierbare digitale signaturen - theorie und praxis. *DuD*, 35(11):757–762, 2011.

[121] R. Steinfeld, L. Bull, and Y. Zheng. Content extraction signatures. In *ICISC*, pages 285–304, 2001.

[122] K. W. Tan and R. H. Deng. Applying Sanitizable Signature to Web-Service-Enabled Business Processes: Going Beyond Integrity Protection. In *ICWS*, pages 67–74, 2009.

[123] Y. Wang, H. Pang, and R. H. Deng. Verifiably encrypted cascade-instantiable blank signatures to secure progressive decision management. *Int. J. Inf. Sec.*, ??(??):???–???, 2017.

[124] Z.-Y. Wu, C.-W. Hsueh, C.-Y. Tsai, F. Lai, H.-C. Lee, and Y. Chung. Redactable Signatures for Signed CDA Documents. *Journal of Medical Systems*, pages 1–14, 2010.

[125] X. Yi. Directed transitive signature scheme. In *CT-RSA*, pages 129–144, 2007.

[126] D. H. Yum, J. W. Seo, and P. J. Lee. Trapdoor sanitizable signatures made easy. In *ACNS*, pages 53–68, 2010.