Sanitizable and Redactable Signatures

Report for MTE7101 (MSE Project I)

Degree course:	Master of Science in Engineering
Author:	Pascal Mainini
Advisor:	Prof. Dr. Reto Koenig
Constituent:	BFH, Research Institute for Security in the Information Society, E-Voting Group
Date:	2019-01-25

Abstract

Besides the well known digital signatures, many other and lesser known schemes exist. Among them are sanitizable and redactable signatures, which allow for modifications in signed messages without breaking the validity of the signature. Since nearly twenty years, they form an interesting and active field of research.

As a preparation for future work, we have investigated those non-classic schemes for digital signatures, focusing on sanitizable and redactable signatures. This document is a summary of our investigations and aims to provide a quick introduction to the topic, as well as to serve as a starting point into the relevant literature.

Contents

1. Introduction

This document summarizes the efforts of the last four months, in which we have tried to map the landscape of digital signature schemes. With possible future work in mind, we focused on *redactable* and *sanitizable* signature schemes, aiming to provide the interested reader with a quick introduction to the topic and a starting point into relevant literature. In order to stay concise, we do not document our research into adjacent literature regarding cryptographic primitives used by those schemes, as well as other topics discussed during that time. The structure of this document is a follows:

In Chapter 2, we provide an overview of digital signature schemes in general and the definition and security notions of classic digital signatures. To position sanitizable and redactable signature schemes in the broader landscape of digital signatures, we then discuss some digital signature schemes with less known properties.

Chapter 3 then specifically covers sanitizable and redactable signatures. For both, it provides formal definitions and a review of the current state of the art. It closes with a unified view on their security notions and the relations between those.

As a short overview of this work, the graphical representation given in Appendix A may be helpful.

Finally, we would like to thank Reto E. Koenig and Rolf Haenni for proposing the topic. Also, we are deeply grateful for the sheer endless amount of fruitful discussions with Reto E. Koenig.

It is impossible for a man to learn what he thinks he already knows. – Epictetus

2. Digital Signature Schemes

Beside well-known 'classic' schemes such as RSA and DSA signatures, a broad range of additional schemes has been developed over the last two decades, supporting manifold application scenarios. In this chapter, we provide an overview and a classification of such schemes, subsumed as *digital signature schemes (DSS)*. It should serve as a high-level map of the DSS landscape, without providing to many details about individual instances. Larger parts of this chapter are based on the extensive review of DSS in [DDH+15], which provide thorough descriptions and underlying definitions.

The classification in this chapter follows [DDH+15], which separates schemes into *functional* and *malleable* (or also *algebraic*) signature schemes (see Section 2.2). Malleable schemes (MS) are further divided into *MS for arithmetics* and *MS for editing*. The authors of [DDH+15] point out, that their classification has been chosen somewhat arbitrarily; however fitting well our purpose, we have also taken it as a base for our own classification.

After presenting functional and malleable schemes, we conclude in Section 2.3 with an overview of further DSS, which we have identified during our research.

A graphical representation of our classification is given in Figure 2.1. Schemes relevant for this document are highlighted.

2.1. Classic Digital Signature Schemes

For easier presentation and understanding of the different or additional properties of the more advanced schemes, we provide a short and informal definition of classic digital signature schemes, based on [KL14, 12.2, p. 441], in this section.

A classic DSS consists of the following three *probabilistic polynomial time (efficient)* algorithms:

1. Algorithm $\mathsf{KeyGen}(1^\lambda)$ which outputs a public-/secret key pair (pk, sk) based on a security parameter λ:

   $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$

2. Algorithm $\mathsf{Sign}(sk, m)$ which signs message m using secret key sk and outputs a signature σ:

   $\sigma \leftarrow \mathsf{Sign}(sk, m)$

3. Algorithm $\mathsf{Verify}(pk, \sigma, m)$ which takes as input the public key pk, signature σ and the message m. It outputs a bit $b \in \{0, 1\}$ with $b = 1$ if the signature is valid and $b = 0$ otherwise:

   $b \leftarrow \mathsf{Verify}(pk, \sigma, m) \qquad b \in \{0, 1\}$

The DSS must fulfill the following (security) notions:

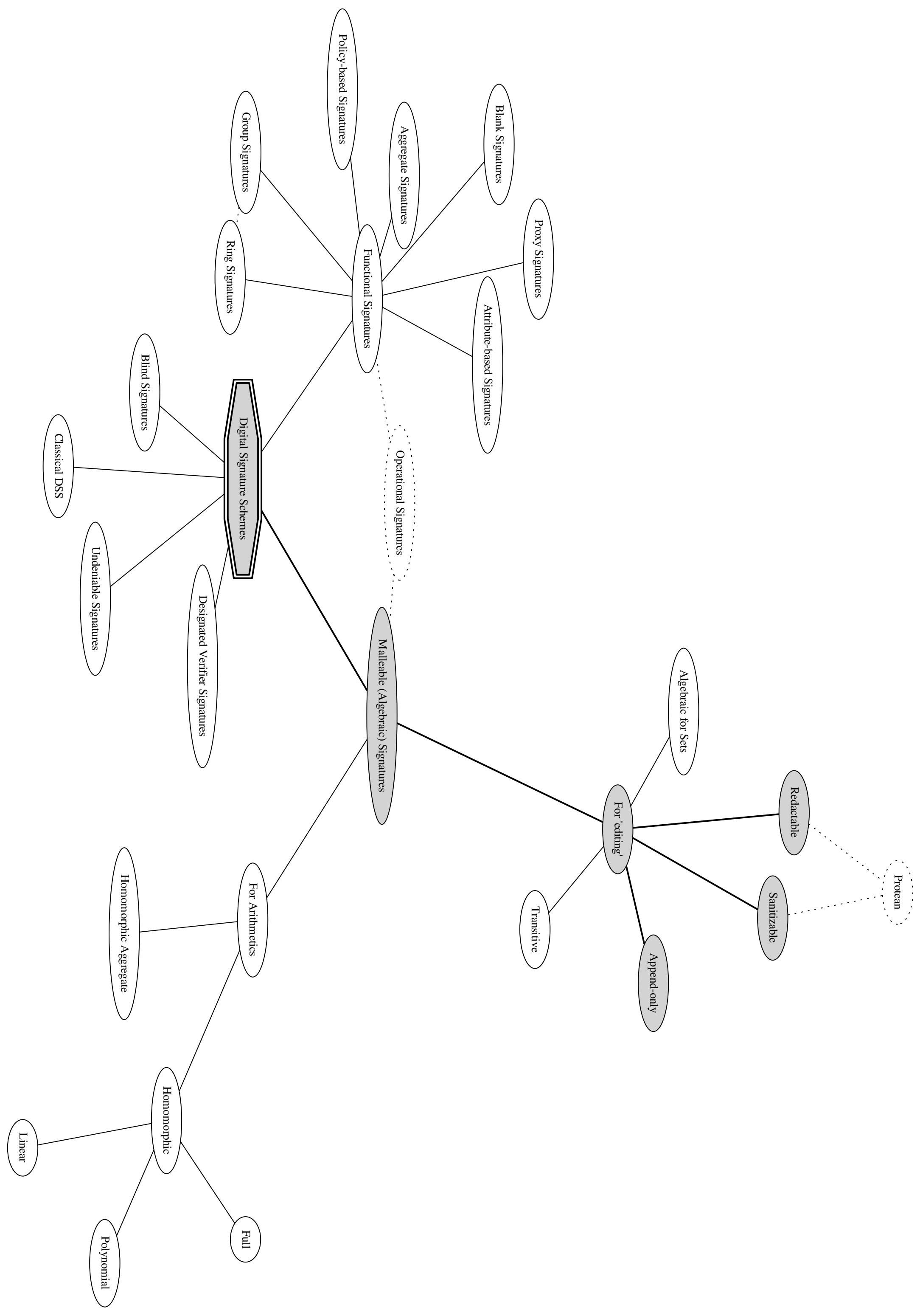

Figure 2.1.: Overview of Digital Signature Schemes

Correctness A DSS is required to be *correct*: Except with negligible probability, it holds that $\mathsf{Verify}(pk, \mathsf{Sign}(sk, m), m) = 1$ for any (pk, sk) output by KeyGen and for any message m.

Unforgeability A DSS is required to be *existentially unforgeable under chosen-message attacks (EUF-CMA)*. A *forgery* is a valid signature for a message and a public key belonging to a given signer, where the message has not been signed by the signer before. EUF-CMA requires, that an adversary is not able to output such a forgery, even if it can obtain signatures by the signer on many messages of its choice.

We note, that DSS often use *hash-then-sign*: Due to limitations in the message space and efficiency of the underlying public key crypto system, typically a hash of the message (taken using a collision resistant hash function) and not the message itself is signed in a DSS.

2.2. Functional and Malleable Signature Schemes

2.2.1. General Frameworks

Functional and malleable signature schemes can roughly be divided into general frameworks, which support the creation of individual functional or malleable schemes, and in concrete instantiations of individual schemes. Due to their generality, the frameworks may be very inefficient and must sometimes only be considered as feasibility results. [DDH$^+$15] identify the following general frameworks:

- *Policy-based signatures.* Here, the message space a signer is allowed to sign, can be restricted by a policy. This may be seen as a special case of functional signatures, where the policy is the function f (see below).
- *P-Homomorphic signatures.* These are signature schemes which allow for public computations on signed data.
- *Malleable signatures for general transformations.* Generalized signature schemes, which allow transformations on message-signature pairs, while keeping the signature valid.
- *Functional signatures.* Allow for additional signing keys which are restricted by a given function f. Signatures using those keys can only be generated for messages m in the range of f, i.e. on $f(m)$.
- *Operational signatures.* Recently introduced as a more general framework, uniting functional and malleable signatures.

We do not further detail those frameworks here and refer to [DDH$^+$15] for exact definitions, as well as concrete constructions. In the following, we present a selection of specific functional and malleable signatures, which we encountered frequently during our research.

2.2.2. Specific Functional Signatures

A well known example of functional signatures are *group signatures* and related to them *ring signatures* and *proxy signatures*. In group signatures, a manager sets up a group with multiple members having individual keys. Each member of the group can then anonymously create signatures, however the group manager can revoke the anonymity using his key. Ring signatures are quite similar to group signatures, however there is no manager and no explicit group setup required. Therefore, there is also no way to revoke anonymity. In proxy signatures, a signer may delegate its signing rights to another party; this delegation may also occur hierarchically.

Another type of functional signatures are *attribute based signatures*, where signing keys only work for messages having specific attributes.

Finally, *blank signatures* are another interesting scheme, which is also somewhat related to proxy signatures. In blank signatures, a signer may delegate signing rights for templates. These templates contain fixed and exchangeable parts. The delegated signer may then sign a message consisting of the fixed parts of the template as well as a single option chosen per exchangeable part.

2.2.3. Specific Malleable Signatures

Malleable signatures can be divided logically into two classes: homomorphic (arithmetic) and malleable for editing.

Homomorphic signatures allow for arithmetic computations on signed messages. They are separated into distinct types following [DDH$^+$15]: linearly homomorphic, polynomial homomorphic and fully homomorphic – depending on the applicable functions. Furthermore, *homomorphic aggregate signatures* support the aggregation of multiple homomorphic signatures (see also Section 2.3.2). It is important to note, that homomorphic signatures have different security notions: due to their nature, existential forgeries are explicitly possible!

Malleable signatures for editing can be separated into different classes as well: *sanitizable*, *redactable* and *append-only*. Append-only schemes allow to publicly append additional blocks to a message and update the signature accordingly. Redactable signatures provide the opposite, allowing to redact (i.e. remove or censor) blocks of a message. In sanitizable schemes, modifications of signed messages may be conducted in a predefined way. Only recently, [KPSS18] have introduced the notion of *protean signatures*, potentially uniting redactable and sanitizable signatures. Furthermore, [DDH$^+$15] present *transitive signatures* and *algebraic signatures for sets* as additional, related concepts, which we do not further detail.

In Chapter 3, we will revisit redactable and sanitizable signatures in more detail and review the current state of the art.

2.3. Further Digital Signature Schemes

Besides functional and malleable signatures, we have identified further types of digital signatures, which we shortly describe in the following.

2.3.1. Blind Signatures

Blind signatures, introduced by Chaum[Cha83, Cha85] are an early extension of classic DSS. Besides Keygen, Sign and Verify, they consist of two additional algorithms:

1. Algorithm Blind which takes the public key of the signer and the message, and returns blinded parameters.
2. Algorithm Unblind which takes the public key of the signer, a blinded signature and the blinded parameters, and returns the unblinded signature.

These two algorithms enable signatures, in which the signer does not know about the contents of the message to be signed. The signature on the later unblinded message is valid and can be verified using the signers public key.

2.3.2. Aggregate Signatures

Aggregate signatures support aggregation of multiple signatures of distinct messages (potentially from different signers) into a single, succinct signature. Verification occurs on the aggregated signature, together with the individual messages. Success indicates, that the distinct original messages have effectively been signed by the distinct users.

2.3.3. Undeniable Signatures

Undeniable signatures enable the signer to choose who can verify a signature. For this, either one of two zero-knowledge protocols is run interactively between signer and verifier. A signature verifies correctly, if the confirmation protocol is executed, or it does not verify if the so-called disavowal protocol is executed.

2.3.4. Designated Verifier Signatures

Designated verifier signatures can be seen as an improvement of undeniable signatures, eliminating the requirement of interactive interaction. Various schemes have been adapted to be *designated verifier* schemes by using a disjunctive proof of knowledge. We further describe this technique when reviewing [DKS16] in Section 3.2.2 (p. 12).

3. Sanitizable and Redactable Signatures

The notions of *sanitizing* and *redacting* might look quite similar at first: it is conceivable to implement redaction as a form of sanitization, where parts to be redacted are simply replaced by a special 'blanking', or NULL symbol. However, already the visibility of a redaction may be considered as a leak in privacy, depending on the content of the redacted data. Due to this, both schemes are generally treated independently in the literature, even though there is some overlap between definitions and security notions, and some schemes might bridge the gap (e.g. [KPSS18]).

In this chapter, we first take a look at sanitizable (SSS) and later at redactable signature schemes (RSS). For both, we provide definitions and an overview of the current state of the art. As an important amount of research has been conducted and is still going on in this field, providing complete coverage proved to be difficult. We thus have tried to find the relevant corner stones for both types of signatures, while focusing more on RSS. This focus is due to our upcoming work, which mainly targets RSS.

3.1. Sanitizable Signature Schemes

Using sanitizable signatures, a signer can delegate signing rights to a *designated sanitizer*. The sanitizer can then sanitize, i.e. modify, (parts of) the message in a way predetermined by the signer. The resulting signature on a properly sanitized message still verifies correctly and can be traced back to the original signer.

3.1.1. Definition

Compared to DSS (2.1, p. 3), SSS require some additional, efficient algorithms. We now give a definition of all algorithms of a SSS, based on [BFF⁺09]:

1. Algorithm $\mathsf{KeyGen}_{\mathsf{sig}}(1^\lambda)$ which outputs a public-/secret key pair (pk_{sig}, sk_{sig}) for the signer, based on a security parameter λ:

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

2. Algorithm $\mathsf{KeyGen}_{\mathsf{san}}(1^\lambda)$ which outputs a public-/secret key pair (pk_{san}, sk_{san}) for the sanitizer, based on a security parameter λ:

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

3. Algorithm $\mathsf{Sign}(sk_{sig}, pk_{san}, m, adm)$ which signs message m using the secret key sk_{sig} of the signer, the public key pk_{san} of the sanitizer and the description of admissible sanitization adm. It outputs a signature σ:

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

4. Algorithm $\mathsf{Sanitize}(pk_{sig}, sk_{san}, m, \sigma, mod)$ which sanitizes a message m using sanitization information mod, the secret key sk_{san} of the sanitizer, the public key pk_{sig} of the signer and the signature σ. It outputs (m', σ') where m' is the sanitized message and σ' the sanitized signature:

   $$(m', \sigma') \leftarrow \mathsf{Sanitize}(pk_{sig}, sk_{san}, m, \sigma, mod)$$

5. Algorithm $\mathsf{Verify}(pk_{sig}, pk_{san}, \sigma, m)$ which takes as input the public key pk_{sig} of the signer, the public key pk_{san} of the sanitizer, signature σ and the message m. It outputs a bit $b \in \{0, 1\}$ with $b = 1$ if the signature is valid and $b = 0$ otherwise:

   $$b \leftarrow \mathsf{Verify}(pk_{sig}, pk_{san}, \sigma, m) \qquad b \in \{0, 1\}$$

6. Algorithm $\mathsf{Proof}(sk_{sig}, pk_{san}, m, \sigma, (m_1, \sigma_1), \ldots, (m_k, \sigma_k))$ which takes as input the secret key sk_{sig} of the signer, the public key pk_{san} of the sanitizer, a message m and signature σ, as well as a set of, say k, additional message-signature pairs $(m_i, \sigma_i)_{i=1,2,\ldots,k}$. It outputs a proof string $\pi \in \{0, 1\}^*$:

   $$\pi \leftarrow \mathsf{Proof}(sk_{sig}, pk_{san}, m, \sigma, (m_1, \sigma_1), \ldots, (m_k, \sigma_k)) \qquad \pi \in \{0, 1\}^*$$

7. Algorithm $\mathsf{Judge}(pk_{sig}, pk_{san}, \sigma, m, \pi)$ which takes as input the public key of the signer pk_{sig}, the public key of the sanitizer pk_{san}, a signature σ, message m and a proof π. It outputs a decision $d \in \{sig, san\}$, indicating which party has created the message-signature pair (signer or sanitizer):

   $$d \leftarrow \mathsf{Judge}(pk_{sig}, pk_{san}, \sigma, m, \pi) \qquad d \in \{sig, san\}$$

The algorithms Proof and Judge are required for the security notion of accountability, which we describe in Section 3.3 below.

3.1.2. State of the Art

Sanitizable signatures were originally formulated by Ateniese et al. [ACdMT05], sometimes also Johnson et al. [JMSW02] is attributed. In their work, a formal definition of SSS is given and the fundamental security notions of *unforgeability*, *immutability*, *privacy*, *transparency* and *accountability* have been introduced. Additionally, [ACdMT05] provide a construction based on *chameleon hashes*, where the basic idea is that admissible parts are hashed using a chameleon hash, while each non-admissible part is hashed using a standard cryptographic hash. Anyone in possession of the secret key for the chameleon hash can then change such a part and efficiently calculate a collision, thus keeping the signature valid.

The security notions of [ACdMT05] have been revisited by Brzuska et al., [BFF$^+$09], which provide stronger definitions and find a contradiction in the relation of unforgeability and accountability. We revisit these notions and their relations in Section 3.3.

Later, again extensions were proposed, which not only include the message and signature, but also include adm ([GQZ11]) or provide a notion of *strong unforgeability* ([KSS16]). Finally, further security notions have been added, especially the notions of *unlinkability* and *invisibility*, introduced in [BFLS10] and [CDK$^+$17] respectively. We do not detail these further.

Recently, a position paper by Bilzhause et al., [BPS17], summarizes the past, present and future of RSS, as well as SSS. It is a concise review of the current state of the art, definitions and research opportunities.

3.2. Redactable Signature Schemes

The notion of *redacting* is that of removing or censoring ('blacking out') parts of a signed message, which is different from sanitization, as stated in the introduction. Opposed to SSS, redactions of a message can in general be conducted publicly, i.e. by any party, requiring only access to the signed message and the public key of the signer. The resulting signature on a properly redacted message then still verifies correctly and can be traced back to the original signer.

RSS provide an interesting primitive for *privacy enhancing/preserving cryptography*: We can think of a variety of use cases (for instance in e-health and e-government), which follow a basic protocol, between three independent parties, similar to the following:

1. A user requests that an authority confirms the authenticity of a message (or more specifically, a set of data). The message could also be provided by the authority without prior request.
2. The message is signed by the authority using a RSS.
3. The user can redact parts of the signed message and forward it to any third party.
4. The third party can verify the (redacted) signature and thus the authenticity of the message without gaining any insight about the redacted parts.

3.2.1. Definition

RSS require only a single additional, efficient algorithm, compared to DSS (2.1, p. 3), namely Redact. We now give a definition of all algorithms of a RSS:

1. Algorithm $\mathsf{KeyGen}(1^\lambda)$ which outputs a public-/secret key pair (pk, sk) based on a security parameter λ:

   $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$

2. Algorithm $\mathsf{Sign}(sk, m, adm)$ which takes as input the secret key of the signer sk, message m and a description of admissible redactions adm. It outputs a signature σ:

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

3. Algorithm $\mathsf{Verify}(pk, \sigma, m)$ which takes as input the public key pk, signature σ and the message m. It outputs a bit $b \in \{0, 1\}$ with $b = 1$ if the signature is valid and $b = 0$ otherwise:

   $b \leftarrow \mathsf{Verify}(pk, \sigma, m) \qquad b \in \{0, 1\}$

4. Algorithm $\mathsf{Redact}(pk, \sigma, m, mod)$ which takes as input the public key of the signer pk, signature σ, message m and some modification instructions mod. It outputs (m', σ') where m' is the redacted message and σ' the redacted signature:

   $(m', \sigma') \leftarrow \mathsf{Redact}(pk, \sigma, m, mod)$

3.2.2. State of the Art

Early and independent initial publications about RSS are given by Steinfeld et al., [SBZ01], and also Johnson et al., [JMSW02].We find some similarities between the proposed constructions, especially the usage of Merkle-trees ([Mer80]) to reduce signature size. The authors also propose initial security notions: While [JMSW02] only defines *unforgeability*, [SBZ01] already introduces some notion of *privacy*.

Later, Brzuska et al., [BBD$^+$10], contribute the first formal definitions for the notions of *privacy* and *transparency* in the context of RSS, after having revisited the security notions of SSS before ([BFF$^+$09]). We describe these notions in Section 3.3 below. As a practical contribution, they present a RSS construction for tree based data, whose concept can be outlined as follows: Using a classic DSS, all edges of a given tree are signed. Also, for all edges, all adjacent vertices are decorated using a distinct random number and signed as well. This prevents attacks in which multiple trees could be matched and mixed. The redactable signature over the tree is then simply the set of all these signatures and redacting is conducted by omitting individual signatures from the set (cf. [BBD$^+$10, Section 5.1] for more details).

The usage of so-called *cryptographic accumulators* has motivated a range of constructions for RSS. [PSPDM12] provide two constructions for unordered sets as well as for ordered lists ('linear documents'). [DPSS15] take the idea further and provide a general framework for RSS, which can be applied to arbitrary data structures. The formalization of the framework is based on [BBD$^+$10], however the Sign and Redact algorithms are extended with structures describing admissible redactions, modification instructions and auxiliary redaction information (see [DPSS15, Section 2.1]. Based on this additional data, the framework can be adjusted to individual data structures. As an example, the authors provide two constructions, again one for sets and one for ordered lists, which are both based on any DSS and cryptographic accumulators. For the *RSS for sets* ([DPSS15, Section 4]), the basic principle is as follows: An accumulator representing the set is computed and then signed using the DSS. For verification, a witness for each element in the set (given) as well as the signature of the accumulator have to be verified. Redaction is performed by throwing away witnesses corresponding to redacted elements. The *RSS for ordered lists* ([DPSS15, Section 5]) is quite similar, however it takes some precautions when encoding positions of elements, as redactions would be trivially identifiable from missing elements in the order. Both schemes support fixed (unredactable) elements defined by the signer and also dependencies between elements.

Normally, RSS do not provide the notion of *accountability*: it cannot be determined, if a signature has been created by the signer or through redaction (transparency notion, see Section 3.3). A formalization of accountability in RSS is presented in [PS15]. The authors distinguish between an on-line- and an off-line form. In the latter, everyone can derive the accountable party (so-called public accountability), however this breaks the transparency notion. A construction based on a combination of a SSS, combined with a RSS is presented, which offers both forms of accountability, depending on the underlying SSS.

Finally, as an example of what can be achieved by combining RSS with other primitives, we describe [DKS16], which is a RSS specifically targeted at a use case in e-health. It offers the interesting properties of *signer anonymity* and verifiability only through *designated verifiers*. The RSS is a black box construction, making use of generic primitives for redactable signatures (e.g. [DPSS15]), group signatures and proof systems for zero-knowledge proofs of knowledge. Signer anonymity is then obtained by generating redactable signatures on a message using an ephemeral RSS key pair, of which the public key is certified using a group signature. Designated verifiability is achieved using a disjunctive proof of knowledge.[1] The authors state that, depending on the underlying RSS, such proofs may be expensive to compute (cf. [DKS16, Section 4]). In order to increase efficiency, they introduce key-homomorphic signatures, which enable a more efficient disjunctive proof of knowledge. As a side effect, the authors also claim to obtain the first *group redactable signature scheme*, which follows simply from omitting the notions related to the designated verifiability.

Again, we refer to [BPS17] and also [DDH$^+$15] for further references of current state of the art.

3.3. Security Notions

We review the security notions of SSS and RSS. Some are shared by both schemes, while SSS require some additional notions. We first describe all notions shared between the two schemes, pointing out important differences. Then, the additional notions of SSS are discussed. We base our discussion on [BFF$^+$09, BBD$^+$10, BPS17]. As for DSS, we also state *correctness*, which is not strictly a security notion.

3.3.1. Common Security Notions

The following notions exist for SSS and RSS equally:

Correctness Given correct parameters, a signature which was computed by a RSS or SSS verifies correctly, under the assumption that all involved parties behave honestly. The same applies also to redacted or sanitized signatures.

Unforgeability This notion is similar to the unforgeability notion for DSS: Without having access to the relevant secret key, the computation of a valid signature on any message is not possible. It must however be adapted to SSS and RSS:

For SSS, a forgery for any party except the signer and the sanitizer must not be possible. Clearly, the sanitizer constructs a deliberate forgery (within admissible bounds) of a message signed by the signer.

For RSS, forging a signature for any message, which is *not* a valid redaction, must be infeasible without access to the secret key of the signer.

Privacy No information about the original message must be obtained from a sanitized or redacted message.

Transparency Signatures on unredacted data must not be distinguishable from signatures obtained by redaction or sanitization; i.e. it must not be decidable from a message and a signature, if the message has been sanitized or redacted.

[1]Non-interactive proofs of knowledge are in general not designated, as the transcript can be verified by anyone using the given values. A well known technique to transform non-interactive proofs into designated proofs is based on a disjunctive ('OR') proof demonstrating either knowledge of a valid signature or knowledge of the verifiers secret key. Such a proof is convincing only to the verifier, because any valid transcript could also have been generated by the verifier itself. This construction is due to Jakobsson, Sako and Impagliazzo, [JSI96].

3.3.2. Security Notions for SSS

These notions only exist for SSS:

Immutability The sanitizer must not be able to create correctly verifying signatures for messages, which contain sanitizations which are not defined as admissible in *adm* (i.e. contain not admissible sanitizations).

Additional sanitizing attacks: This is an attack, in which a sanitized message is again sanitized by a different, malicious sanitizer. Assuming different keys per sanitizer, the notion given in [BFF+09] still holds in this case.

Accountability Neither the signer nor the sanitizer should be held responsible for signatures coming from the other party. Using Proof, the signer can generate proofs which convince Judge that a signature has not been created by itself (or vice versa). This also requires, that neither signer nor sanitizer can make up *false* proofs.

3.3.3. Relations Between Notions

The security notions described above are not completely independent and have some relations to each other, which are depicted in Figure 3.1. The definition of these relations is also due to [BFF+09, BBD+10]. We now describe the relations for SSS, the relations of privacy and transparency similarly apply to RSS.

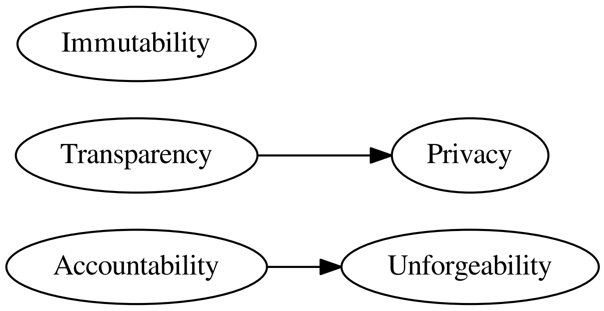

Figure 3.1.: Relations Between Security Notions

The first relation implies that *privacy follows from transparency*. The relation is derived from the proof of privacy in [BFF+09], in which an adversary may submit two message-modification tuples and an oracle returns a sanitized signature on one of them.The adversary must then not be able to decide, which of the input tuples has been sanitized (clearly, for this the sanitized messages must be identical). Transparency states, that signatures of the signer and sanitizer must be indistinguishable, thus the oracle algorithm may be changed to use Sign instead of Sanitize. However, as the proof of privacy requires the sanitized messages to be identical, the signatures must always be different for the same message. *We note that transparency is a stronger notion than privacy, as not only the structure and content of a message is protected, but also if a sanitization or redaction has been performed.*

The second relation implies that *unforgeability follows from accountability* (thereby contradicting [ACdMT05], possibly due to separate/different consideration of signer-accountability and sanitizer-accountability). The idea here is to provide Judge with a forgery as input. Judge then cannot decide, which party to account for the signature.

Finally, using additional proofs which we do not detail, [BFF$^+$09] show that there are no further relations between the security notions. The basic idea of these proofs is to start from a SSS which is *secure*, i.e. has all given notions, and to then construct multiple modified versions of it, which are missing a single, separate notion every time.

4. Conclusion and Outlook

During our research, we were able to discover a larger part of the landscape of digital signature schemes. Our focus on sanitizable and redactable schemes enabled us to review most relevant milestones in that field of research, which spans over a period of almost the last twenty years. We have written down our insights in this report and presented them at Bern University of Applied Sciences in the course of this project. Deeply thinking about SSS and RSS, during writing and many discussions, made a lot of things clearer to us.

We are confident, that this report will serve us well as a reference for the next steps to come: We look forward to gain further insights by applying the acquired knowledge in the implementation of a redactable, maybe also sanitizable signature scheme for a concrete, yet to be defined use case. We also hope to devise our own construction to further deepen our knowledge and maybe even actively participate in ongoing research.

Glossary

Chameleon Hash Chameleon hashes are a special type of hash function. They enable efficient calculation of collisions if a secret (or trapdoor) is known..

Digital Signature Scheme (DSS) Generally refers to any classic, well known digital signature scheme, as for instance the digital signature algorithm (DSA). In this document, DSS refers to *all* kinds of signature schemes, including non-classic as described in Chapter 2 (p. 3).

EUF-CMA EUF-CMA is a security definition for digital signature schemes, which signifies '*existential unforgeability under chosen message attacks*'. A short description is given in Section 2.1 (p. 3).

Hash Function Hash functions map arbitrary sized data to a single, succinct value. We especially distinguish *cryptographic* hash functions which require additional properties, like for instance being collision-resistant.

Primitive (Cryptographic) A cryptographic primitive is a building block, which can be used in schemes and protocols to achieve a certain functionality, e.g. functions for encrypting or hashing data.

Protocol (Cryptographic) A cryptographic protocol can be considered as algorithm running between multiple parties. An example would be interactive zero-knowledge proofs of knowledge. Protocols may lead to a result on their own or serve as primitive for other uses.

Redactable Signature Scheme (RSS) See detailed introduction in Chapter 3 (p. 9).

Sanitizable Signature Scheme (DSS) See detailed introduction in Chapter 3 (p. 9).

Scheme (Cryptographic) A cryptographic scheme is a defined way to assemble cryptographic primitives to build more complex functions. An example are the digital signature schemes introduced in Chapter 2 (p. 3).

Zero Knowledge Proof of Knowledge Zero Knowledge Proofs of Knowledge enable a party to prove to another party some specific knowledge, without disclosing the actual knowledge. This may be done in an interactive fashion, where the two parties interact with each other, or in a non-interactive way, where the proof can be calculated without interaction.

Bibliography

[ACdMT05] Giuseppe Ateniese, Daniel Chou, Breno de Medeiros, and Gene Tsudik, *Sanitizable signatures*, 09 2005, pp. 159–177.

[BBD$^+$10] Christina Brzuska, Heike Busch, Oezguer Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, and Dominique Schröder, *Redactable signatures for tree-structured data: Definitions and constructions*, Proceedings of the 8th International Conference on Applied Cryptography and Network Security (Berlin, Heidelberg), ACNS'10, Springer-Verlag, 2010, pp. 87–104.

[BFF$^+$09] Christina Brzuska, Marc Fischlin, Tobias Freudenreich, Anja Lehmann, Marcus Page, Jakob Schelbert, Dominique Schröder, and Florian Volk, *Security of sanitizable signatures revisited*, Public Key Cryptography – PKC 2009 (Berlin, Heidelberg) (Stanisław Jarecki and Gene Tsudik, eds.), Springer Berlin Heidelberg, 2009, pp. 317–336.

[BFLS10] Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder, *Unlinkability of sanitizable signatures*, Public Key Cryptography – PKC 2010 (Berlin, Heidelberg) (Phong Q. Nguyen and David Pointcheval, eds.), Springer Berlin Heidelberg, 2010, pp. 444–461.

[BPS17] Arne Bilzhause, Henrich C. Pöhls, and Kai Samelin, *Position paper: The past, present, and future of sanitizable and redactable signatures*, Proceedings of the 12th International Conference on Availability, Reliability and Security (New York, NY, USA), ARES '17, ACM, 2017, pp. 87:1–87:9.

[CDK$^+$17] Jan Camenisch, David Derler, Stephan Krenn, Henrich C. Pöhls, Kai Samelin, and Daniel Slamanig, *Chameleon-hashes with ephemeral trapdoors*, Public-Key Cryptography – PKC 2017 (Berlin, Heidelberg) (Serge Fehr, ed.), Springer Berlin Heidelberg, 2017, pp. 152–182.

[Cha83] David Chaum, *Blind signatures for untraceable payments*, Advances in Cryptology (Boston, MA) (David Chaum, Ronald L. Rivest, and Alan T. Sherman, eds.), Springer US, 1983, pp. 199–203.

[Cha85] ______, *Security without identification: Transaction systems to make big brother obsolete*, Commun. ACM **28** (1985), no. 10, 1030–1044.

[DDH$^+$15] Denise Demirel, David Derler, Christian Hanser, Heinrich C. Pöhls, Daniel Slamanig, and Giulia Traverso, *Prismacloud d4.4: Overview of functional and malleable signature schemes. technical report, h2020 prismacloud,*, Tech. report, H2020 Prismacloud, 2015.

[DKS16] David Derler, Stephan Krenn, and Daniel Slamanig, *Signer-anonymous designated-verifier redactable signatures for cloud-based data sharing*, Cryptology ePrint Archive, Report 2016/1064, 2016, `https://eprint.iacr.org/2016/1064`.

[DPSS15] David Derler, Henrich C. Pöhls, Kai Samelin, and Daniel Slamanig, *A general framework for redactable signatures and new constructions*, Cryptology ePrint Archive, Report 2015/1059, 2015, `https://eprint.iacr.org/2015/1059`.

[GQZ11] Junqing Gong, Haifeng Qian, and Yuan Zhou, *Fully-secure and practical sanitizable signatures*, Information Security and Cryptology (Berlin, Heidelberg) (Xuejia Lai, Moti Yung, and Dongdai Lin, eds.), Springer Berlin Heidelberg, 2011, pp. 300–317.

[JMSW02] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner, *Homomorphic signature schemes*, Proceedings of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology (London, UK, UK), CT-RSA '02, Springer-Verlag, 2002, pp. 244–262.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo, *Designated verifier proofs and their applications*, Advances in Cryptology — EUROCRYPT '96 (Berlin, Heidelberg) (Ueli Maurer, ed.), Springer Berlin Heidelberg, 1996, pp. 143–154.

[KL14] Jonathan Katz and Yehuda Lindell, *Introduction to Modern Cryptography, Second Edition*, 2nd ed., Chapman & Hall/CRC, 2014.

[KPSS18] Stephan Krenn, Henrich C. PÃ¶hls, Kai Samelin, and Daniel Slamanig, *Protean signature schemes*, Cryptology ePrint Archive, Report 2018/970, 2018, `https://eprint.iacr.org/2018/970`.

[KSS16] Stephan Krenn, Kai Samelin, and Dieter Sommer, *Stronger security for sanitizable signatures*, Data Privacy Management, and Security Assurance (Cham) (Joaquin Garcia-Alfaro, Guillermo Navarro-Arribas, Alessandro Aldini, Fabio Martinelli, and Neeraj Suri, eds.), Springer International Publishing, 2016, pp. 100–117.

[Mer80] Ralph Merkle, *Protocols for public key cryptosystems*, 04 1980, pp. 122–134.

[PS15] Henrich C. Pöhls and Kai Samelin, *Accountable redactable signatures*, Proceedings of the 2015 10th International Conference on Availability, Reliability and Security (Washington, DC, USA), ARES '15, IEEE Computer Society, 2015, pp. 60–69.

[PSPDM12] Henrich C. Poehls, Kai Samelin, Joachim Posegga, and Hermann De Meer, *Length-hiding redactable signatures from one-way accumulators in o(n)*, Tech. Report MIP-1201, Faculty of Computer Science and Mathematics (FIM), University of Passau, 2012.

[SBZ01] Ron Steinfeld, Laurence Bull, and Yuliang Zheng, *Content extraction signatures*, In International Conference on Information Security and Cryptology ICISC 2001, volume 2288 of LNCS, Springer-Verlag, 2001, pp. 285–304.

APPENDICES

A. Appendix A: Presentation Contents

This appendix contains the presentation held at Bern University of Applied Sciences on January 25th, 2019. The image A.1 was presented in animated form using *Sozi*[1], the animation may be obtained from the author upon request.

[1] `http://sozi.baierouge.fr/`

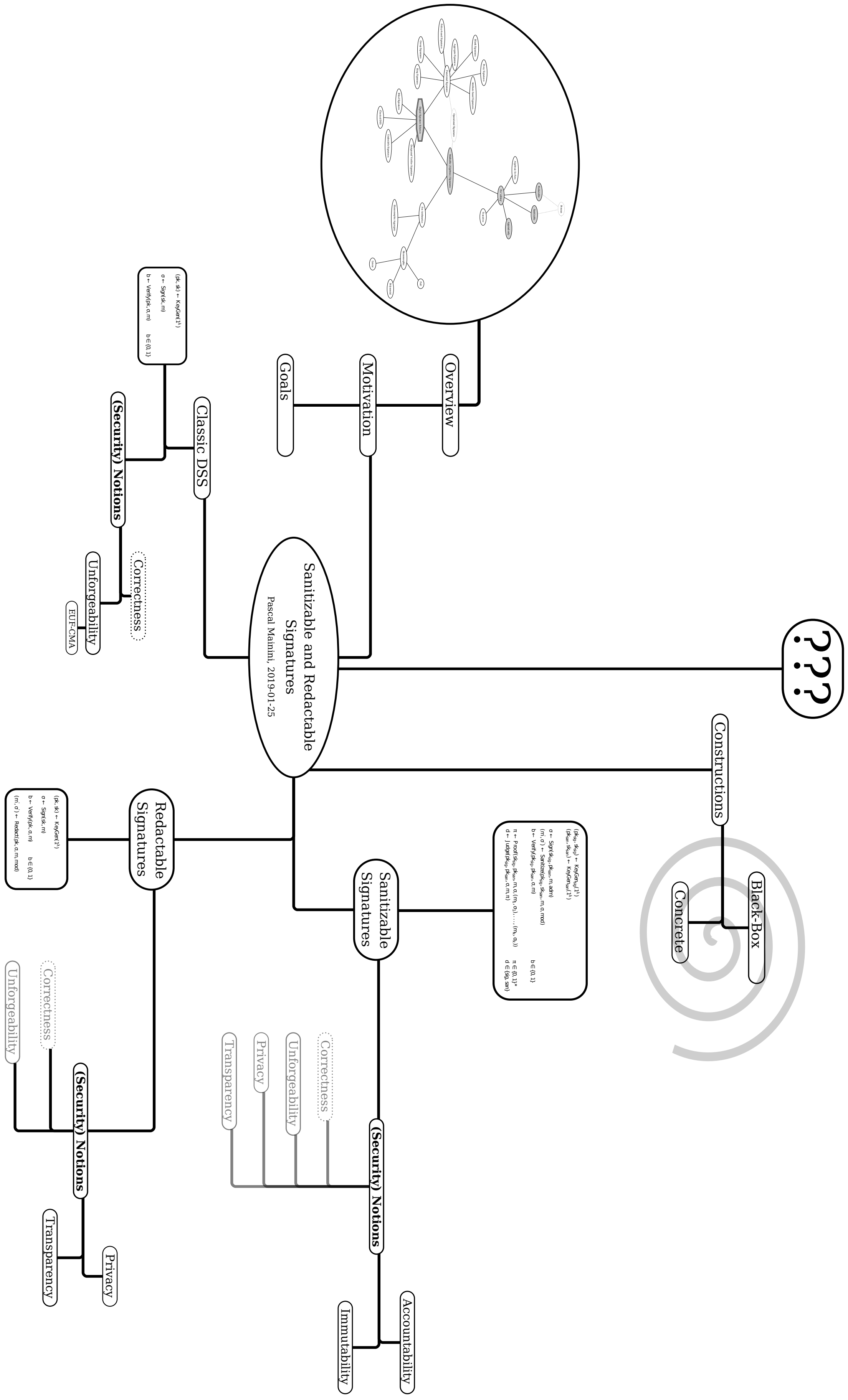

Figure A.1.: Presentation contents 2019-01-25