prisma cloud

Overview of Functional and Malleable Signature Schemes

D4.4

Document Identification	
Date	July 31, 2015
Status	Final
Version	1.0

Related WP	WP4	**Document Reference**	prismacloud.eu
Related Deliverable(s)	-	**Dissemination Level**	PU
Lead Participant	TUG	**Lead Author**	Daniel Slamanig (TUG) David Derler (TUG)
Contributors	Daniel Slamanig (TUG) Christian Hanser (TUG) David Derler (TUG) Denise Demirel (TUDA) Giulia Traverso (TUDA) Henrich C. Pöhls (UNI PASSAU)	**Reviewers**	Helmut Aschbacher (XiTrust) Lucas Schabhüser (TUDA)

D4.4: Overview of Functional and Malleable Signature Schemes

Denise Demirel[†], David Derler[‡], Christian Hanser[‡], Henrich C. Pöhls[§], Daniel Slamanig[‡], and Giulia Traverso[†]

[†] Technische Universität Darmstadt, Germany
[‡] Graz University of Technology, Austria
[§] University of Passau, Germany

Abstract. In this deliverable (D4.4), we investigate the state-of-the-art of specific variants of digital signature schemes, namely functional and malleable signatures. Such signature schemes allow to delegate signature generation for specific types of messages to other parties and allow to modify already signed messages in a controlled way, without signer-interaction (access to the secret signing key), while preserving the validity of the original signature respectively. Especially these schemes seem to be a promising cryptographic tool to provide integrity, authenticity and verifiability in context of outsourced processing as it is widely encountered within cloud computing.

This document is issued within the frame and for the purpose of the PRISMACLOUD project. This project has received funding from the European Union's Horizon 2020 Programme for research, technological development and demonstration under grant agreement no. 644962.

Executive Summary

Cloud computing is about outsourcing of storage to and processing of data at third party infrastructure. Besides many obvious benefits, such a paradigm clearly comes with many security and privacy related problems, as cloud providers cannot be considered fully trustworthy or immune to attacks or faults. While problems related to the confidentiality of data in this setting are well recognised and already addressed today by commercial products, there are still many open problems when it comes to the integrity and authenticity of processed data (and outsourced processes in general) as well as the verifiability of data and the processing tasks. The main question in this context is *how* one can ensure that the cloud *works* as required and how we can hold the cloud *accountable* if this is not the case, e.g., if faults happen or if there occurs some malicious deviation from a specified process.

Within the PRISMACLOUD project we are (among others) interested in designing tools that counter problems related to integrity, authenticity and verifiability in the context of cloud computing. Digital signature schemes are a well known tool for guaranteeing the aforementioned properties in practice. However, their practical application is mainly tailored towards the use with documents so far. Additionally, when considering the outsourcing of processes and their integrity, authenticity and verifiability features, conventional digital signatures are not sufficient. Firstly, the document-oriented view may be too coarse-grained as well as too inexpressive and more importantly the dynamicity required when processing data imposes requirements that are often *diametral* to conventional digital signatures. In particular, any manipulation of signed data destroys corresponding signatures and thus prevents (controlled) updates of information.

There are very promising but not yet well known variants of digital signature schemes that can help to overcome the aforementioned issues. In particular, so called *functional* and *malleable* signature schemes are the most promising ones candidates. Thereby, functional signature schemes allow to delegate signature generation for specific types of messages to other parties. Malleable signatures allow to modify already signed messages in a controlled way and without signer-interaction (access to the secret signing key), while preserving the validity of the original signature.

The main purpose of this deliverable (D4.4) is to present the state-of-the-art of the two aforementioned classes of signature schemes.

Document information

Contributors

Name	**Partner**
Daniel Slamanig	(TUG)
David Derler	(TUG)
Christian Hanser	(TUG)
Denise Demirel	(TUDA)
Giulia Traverso	(TUDA)
Henrich C. Pöhls	(UNI PASSAU)

History

0.01	2015-03-05	Daniel Slamanig	1st Draft
0.02	2015-05-04	Daniel Slamanig	Structure
0.03	2015-06-09	Denise Demirel, Giulia Traverso	Section 4
0.04	2015-06-11	Daniel Slamanig, David Derler, Christian Hanser	Section 2
0.05	2015-06-15	David Derler	Section 3
0.06	2015-06-16	Daniel Slamanig	Section 1
0.07	2015-06-18	Henrich C. Pöhls	Section 5
0.08	2015-07-07	Daniel Slamanig	Update Section 1 & 3, Acronyms, Abstract
0.09	2015-07-08	Daniel Slamanig	Executive Summary
0.10	2015-07-10	Daniel Slamanig, David Derler	Update Parts of Section 5
0.11	2015-07-27	Daniel Slamanig, David Derler, Henrich C. Pöhls	Incorporate Reviewer Comments
1.0	2015-07-31	Daniel Slamanig, David Derler, Henrich C. Pöhls	Document Finalization

List of Acronyms

ABGS	Attribute-Based Group Signatures
ABS	Attribute-Based Signatures
AOS	Append-Only Signatures
AoK	Argument of Knowledge
ARSS	Accountable Redactable Signature Scheme
BDS	Blank Digital Signatures
CDH	Computational Diffie-Hellman
DAC	Delegatable Anonymous Credentials
DDH	Decisional Diffie-Hellman
DL	Discrete Logarithm
DLIN	Decisional Linear
DSS	Digital Signature Scheme
ESSS	Extended Sanitizable Signature Scheme
EUF-CMA	Existential Unforgeability under Adaptively Chosen Message Attacks
FE	Functional Encryption
FlexDH	Flexible Diffie-Hellman
FS	Functional Signatures
GS	Groth Sahai
HIBE	Hierarchical Identity-Based Encryption
HIBS	Hierarchical Identity-Based Signatures
ISIS	Inhomogeneous Small Integer Solution
MAC	Message Authentication Code
MS	Malleable Signatures
NIWI	Non-Interactive Witness Indistinguishability
NIZK	Non-Interactive Zero Knowledge
NP	Nondeterministic Polynomial Time
OS	Operational Signatures
OWF	One Way Function
PBS	Policy-Based Signature
PoK	Proof of Knowledge
PPT	Probabilistic Polynomial Time
PRF	Pseudorandom Function
PRG	Pseudorandom Generator
PS	Proxy Signatures
q-SDH	q-Strong Diffie-Hellman
q-SFP	q-Simultaneous Flexible Pairing
RSS	Redactable Signature Scheme
SDP	Simultanous Double Pairing
SE	Simulation Extractable
SIS	Small Integer Solution
SNARK	Succinct Non Interactive Argument of Knowleddge
SSS	Sanitizable Signature Scheme
ZK	Zero Knowledge

List of Figures

List of Tables

Table of Contents

1 Introduction

Outsourcing of storage to and processing of data at third party cloud infrastructure—or more generally, outsourcing services to third parties—introduces various challenging security and trust issues, which need to be solved in order to gain user's trust. Besides the evident confidentiality issues associated with outsourced data, other very important topics are integrity, authenticity and verifiability. Basically, here the question is how we can ensure that the cloud *works* as it is intended or claimed to do and how can we hold the cloud *accountable* if not.

One fundamental cryptographic building block used to provide a means for integrity protection as well as authenticity are digital signatures. A shift from the protection of integrity and authenticity of documents, as traditionally considered by digital signatures, to dynamic processes in a cloud environment introduces additional challenges. Consequently, one requires more versatile and flexible tools to cope with the associated requirements.

Over the years various powerful *variants* of digital signature schemes have been proposed. Among these proposals are so called *malleable* as well as *functional* signature schemes. Thereby, malleable signatures allow some controlled modifications of signed messages that *do not* require the signer's secret key and *do not* destroy the validity of the digital signature. Thus, such schemes can be used to compute on signed (authenticated) data as well as to add accountability to business processes (workflows). Functional signatures allow *delegation* of signature generation to other parties for a class of messages meeting certain conditions (specified by the delegation). Thus, such schemes can be used, among others, to certify computations and processes.

1.1 Motivation

To date, there is no comprehensive study of malleable and functional signature schemes available[1]. As these two classes of digital signature schemes are the most promising for the applications within the PRISMACLOUD project, our goal for this report is to provide an extensive survey of the state-of-the-art within these fields. Moreover, such a complete and structured overview of all the different already available schemes, their potential as well as limitations will help to unleash the potential of such schemes for the application of integrity, authenticity and verifiability as envisioned within PRISMACLOUD.

1.2 Roadmap

Before we begin, we provide a brief overview of the contents of this report. Foremost, in Section 1.3 we provide some preliminary and notational material to make the document self contained. Nevertheless, we want to emphasize that we do not provide an exhaustive introduction to cryptographic primitives such as digital signatures but assume some general familiarity with concepts of modern cryptography.

[1] The first one that covers the part on homomorphic signatures [TDB15] has recently been made available online. It is a result from work within the PRISMACLOUD project and part of this deliverable (cf. Section 4).

The subsequent sections of this report are structured as follows:

General Frameworks: In Section 2, we review general/generic frameworks for functional and malleable/homomorphic signature schemes. We note that some of these frameworks have been proposed independently and cover very similar concepts. Existing generic instantiations for some of these frameworks can only be considered as feasibility results, whereas there are (very) efficient instantiations that fall within some of the frameworks[2]. Such general frameworks are mainly useful to investigate the power as well as limitations of schemes and to investigate connections to other cryptographic primitives (in a black-box way). As we are not going to study such aspects in this report, these frameworks will not appear often throughout the remaining document and do not play any central role in the remaining sections.

Functional Signatures: In Section 3, we review the state-of-the-art in functional signature schemes. Under this class of signature schemes we consider schemes that support functional signing keys, i.e., delegating signing keys that somehow include a functionality which limits its use in which or when/how messages can be signed. Thereby, we cover proxy signature schemes, policy-based signature schemes, attribute-based signature schemes as well as related concepts such as group signatures and traceable signatures.

Malleable Signatures for Arithmetics: In Section 4, we review the state-of-the-art in signature schemes that allow to (arithmetically) compute on signed messages whereas messages are considered as (numeric) data. In particular, we investigate homomorphic signatures schemes that allow to evaluate certain classes of admissible functions (e.g., linear functions, polynomial functions of higher degree) on signed data as well homomorphic aggregate signature schemes.

Malleable Signatures for Editing: In Section 5, we review the state-of-the-art in signature schemes that allow editing signed messages in certain ways without invalidating the signature. Thereby, we have chosen the term *editing*, as these schemes are typically document oriented (instead of data oriented) and their purpose is controlled manipulation of signed documents. This class covers various types of redactable and sanitizable signature schemes as well as append-only signature schemes. We also briefly discuss related concepts such as transitive signatures and graph-based algebraic signatures.

We want to emphasize that the above structure and in particular the classification of the approaches is somewhat arbitrary, not very strict and there are definitely overlaps. We sketch the big picture in Figure 1, where the question mark indicates that there may be other classes which we do not cover here (we stress that the area in the figure does not correlate with the significance of the concepts or numbers of existing schemes).

Most importantly, we have decided to spilt malleable signatures into two separate categories of *malleable signatures for arithmetics* and *malleable signatures for editing*. We do not see any problems with this, as there does not seem to be a very clean separation of the studied

[2]We stress that we provide an overview of all the existing general frameworks without any judging or suggestions which framework may be superior (in case they cover identical or similar functionality).

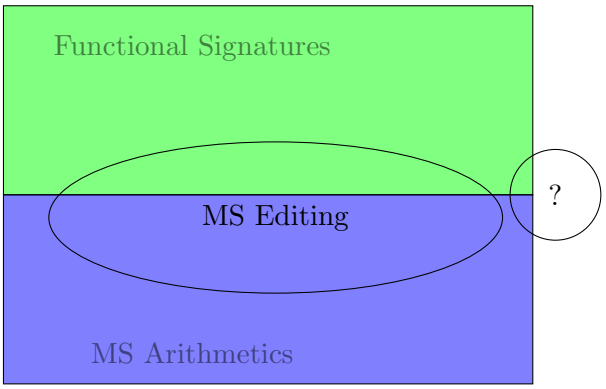

Figure 1: Classification of types of signatures.

approaches (apart from general frameworks) and we believe that our classification is intuitive and more closely reflects the current applications of the studied signature schemes. We also note that in the literature one sometimes encounters the term algebraic signatures instead of homomorphic/malleable signatures, but we will not use it throughout this report. Moreover, we emphasize that we use the term functional signatures in a very broad sense and our class of functional signature schemes may cover more schemes than the *framework of functional digital signatures* discussed in Section 2.1[3]

Finally, in Section 6 we provide an outlook on future research work within the PRISMACLOUD project and general open issues for future work within the various categories of schemes.

1.3 Preliminaries

In the following we introduce some notational conventions that are used throughout the report, some (basic) cryptographic assumptions as well as some basic cryptographic primitives.

1.3.1 Notation

Let $x \xleftarrow{R} X$ denote the operation that picks an element x uniformly at random from a set X. If we write $b \xleftarrow{R} \mathsf{B}(a_1, \ldots, a_\ell)$ where B is an algorithm run on input $a_1, \ldots, a_\ell$, then we emphasize that B is a probabilistic algorithm. Sometimes we write $\mathsf{B}(a_1, \ldots, a_\ell; r)$ to make the randomness r used by a probabilistic algorithm B explicit. A function $\epsilon : \mathbb{N} \to \mathbb{R}^+$ is called *negligible* if for all $c > 0$ there is a k_0 such that $\epsilon(k) < 1/k^c$ for all $k > k_0$. In the remainder of this report, we use ϵ to denote such a negligible function. We will denote a cyclic (additive) group with $\mathbb{G}$ and we use $\mathbb{G}^*$ to denote $\mathbb{G} \setminus \{0_\mathbb{G}\}$. Unless stated differently, we will assume that a group $\mathbb{G}$ has prime order p.

[3]The framework has a very generic name, but not all signatures that fall under functional signatures seem to fit to this framework (for instance, attribute-based signatures do not nicely fit to this framework).

1.3.2 Cryptographic Assumptions

Definition 1 (Bilinear Map). *Let $\mathbb{G}_1 = \langle g \rangle$, $\mathbb{G}_2 = \langle \hat{g} \rangle$ and $\mathbb{G}_T$ be cyclic groups of prime order p, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are additive and $\mathbb{G}_T$ is multiplicative. We call $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ a bilinear map or pairing if it is efficiently computable and the following conditions hold:*

Bilinearity: $e(g^a, \hat{g}^b) = e(g, \hat{g})^{ab} = e(g^b, \hat{g}^a) \quad \forall\, a, b \in \mathbb{Z}_p$

Non-degeneracy: $e(g, \hat{g}) \neq 1_{\mathbb{G}_T}$, *i.e.,* $e(g, \hat{g})$ *generates* $\mathbb{G}_T$.

If $\mathbb{G}_1 = \mathbb{G}_2$, then e is *symmetric* (Type-1) and *asymmetric* (Type-2 or 3) otherwise. For Type-2 pairings there is an efficiently computable isomorphism $\Psi\colon \mathbb{G}_2 \to \mathbb{G}_1$; for Type-3 pairings no such isomorphism is known. Type-3 pairings are currently the optimal choice in terms of efficiency and security trade-off [CM11]. We emphasize that we abuse the multiplicative notion of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ (as often done in the literature).

Now, we state the computational and the decisional version of the Diffie-Hellman problem, where we use the shorthand $\mathcal{G}^\kappa = (\mathbb{G}, p, g)$ with p being a prime of size κ bits.

Definition 2 (CDH). *The computational Diffie-Hellman (CDH) assumption states that for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that*

$$\mathsf{Pr}\left[x, y \xleftarrow{R} \mathbb{Z}_p, \quad \mathcal{A}(\mathcal{G}^\kappa, g^x, g^y) = g^{xy}\right] \leq \epsilon(\kappa).$$

Definition 3 (DDH). *The decisional Diffie-Hellman (DDH) assumption states that for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that*

$$\mathsf{Pr}\left[\; x, y, z \xleftarrow{R} \mathbb{Z}_p, \quad b \xleftarrow{R} \{0,1\}, b^* \leftarrow \mathcal{A}(\mathcal{G}^\kappa, g^x, g^y, g^{(1-b)\cdot z + b\cdot xy}) : b = b^* \;\right] \leq \frac{1}{2} + \epsilon(\kappa).$$

As in certain bilinear groups the decisional Diffie-Hellman problem is easy to solve, e.g,. in Type-1 groups, Boneh et al. [BBS04] introduced the decisional linear assumption which is assumed to hold even in such DDH-easy groups:

Definition 4 (DLIN). *The decisional linear (DLIN) assumption states that for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that*

$$|\mathsf{Pr}[x, y, z \xleftarrow{R} \mathbb{Z}_p, \quad b \xleftarrow{R} \{0,1\}, b^* \leftarrow \mathcal{A}(\mathcal{G}^\kappa, u^x, v^y, h^{(1-b)\cdot z + b\cdot(x+y)}) : b = b^*] \leq \frac{1}{2} + \epsilon(\kappa),$$

where u, v, h are independent generators of $\mathbb{G}$.

The previous problem implies the following simultaneous double pairing problem [CLY09]. Therefore, let $\mathcal{G}^\kappa = (e, \mathbb{G}, \mathbb{G}_T, p, g)$ be a symmetric bilinear group setting with p being a prime of κ bits.

Definition 5 (SDP). *The simultaneous double pairing (SDP) assumption states that for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that*

$$\mathsf{Pr}\left[\begin{array}{c} g_z, h_z, g_r, h_u \xleftarrow{R} \mathbb{G}^*, \quad (z, r, u) \leftarrow \mathcal{A}(\mathcal{G}^\kappa, g_z, h_z, g_r, h_u) : \\ (z, r, u) \in \mathbb{G}^* \quad \wedge \quad e(h_z, z) \cdot e(h_u, u) = 1_{\mathbb{G}_t} = e(g_z, z) \cdot e(g_r, r) \end{array}\right] \leq \epsilon(\kappa).$$

Below we present a variant of the flexible Diffie-Hellman (FlexDH) assumption [KP06], which is slightly stronger than the standard Diffie-Hellman assumption, and is used in [ALP13].

Definition 6 (FlexDH)**.** *The flexible Diffie-Hellman (FlexDH) assumption states that for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that:*

$$\mathsf{Pr}\left[\ a, b \xleftarrow{R} \mathbb{Z}_p, \quad (u, v, w) \leftarrow \mathcal{A}(\mathcal{G}^\kappa, g^a, g^b) : (u, v, w) = (g^\mu, g^{\mu a}, g^{\mu a b}) \quad \wedge \quad \mu \neq 0 \ \right] \leq \epsilon(\kappa).$$

Now we present the non-static q-simultaneous flexible pairing assumption [AFG+10].

Definition 7 (q-SFP)**.** *The q-simultaneous flexible pairing (q-SFP) assumption states that for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that:*

$$\mathsf{Pr}\left[\begin{array}{c} g_z, h_z, g_r, h_u \xleftarrow{R} \mathbb{G}^*, a, \overline{a}, b, \overline{b} \xleftarrow{R} \mathbb{G}, (R_j = (z_j, r_j, s_j, t_j, u_j, v_j, w_j))_{j=1}^q \xleftarrow{R} (\mathbb{G}^7)^q \\ s.t.\ \ each\ \ R_j\ \ satisfies\ \ (1)\ \ and\ \ (2), \\ (z^*, r^*, s^*, t^*, u^*, v^*, w^*) \leftarrow \mathcal{A}(\mathcal{G}^\kappa, g_z, h_z, g_r, h_u, a, b, \overline{a}, \overline{b}, (R_j)_{j=1}^q) : \\ (z^*, r^*, s^*, t^*, u^*, v^*, w^*)\ \ satisfies\ \ (1)\ \ and\ \ (2) \quad \wedge \quad z^* \neq 1 \quad \wedge \quad \forall j \in [q] \quad z^* \neq z_j \end{array}\right] \leq \epsilon(\kappa)$$

where (1) and (2) are defined as:

(1) $e(a, \overline{a}) = e(g_z, z_j) \cdot e(g_r, r_j) \cdot e(s_j, t_j)$,

(2) $e(b, \overline{b}) = e(h_z, z_j) \cdot e(h_r, u_j) \cdot e(v_j, w_j)$,

In the following let $\mathcal{G}^\kappa = (e, \mathbb{G}_1, \mathbb{G}_2\mathbb{G}_T, p, g_1, g_2)$ be a asymmetric bilinear group. We define a Type-2 analogon to the CDH assumption denoted as co-computational Diffie-Hellman (co-CDH) assumption.

Definition 8 (co-CDH)**.** *Let $\mathcal{G}^\kappa$ be an asymmetric Type-2 bilinear group. For all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that given*

$$\mathsf{Pr}\left[a, b \xleftarrow{R} \mathbb{Z}_p, \mathcal{A}(\mathcal{G}^\kappa, g_1, g_1^a, g_2, g_2^b) = g_2^{ab}\right] \leq \epsilon(\kappa)$$

We also state the q-Strong Diffie-Hellman assumption in asymmetric bilinear groups [BB04, BB08].

Definition 9 (q-SDH)**.** *Let $\mathcal{G}^\kappa$ be an asymmetric Type-2 bilinear group, $\alpha \xleftarrow{R} \mathbb{Z}_p^*$ and $q > 0$. For all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that*

$$\mathsf{Pr}\left[\mathcal{A}(\mathcal{G}^\kappa, g_1, g_2, g_2^\alpha, \ldots, g_2^{\alpha^q}) = \left(c, g_1^{\frac{1}{c+\alpha}}\right)\right] \leq \epsilon(\kappa)$$

for $c \in \mathbb{Z}_p \setminus \{-\alpha\}$

Note that due to the efficiently computable isomorphism $\Psi\colon \mathbb{G}_2 \to \mathbb{G}_1$, all the elements are also implicitly available in $\mathbb{G}_1$.

We now present five further complexity assumptions used in [AL11, ALP12].We note that they are defined in a bilinear group setting with composite order groups. Therefore, let $\mathcal{G}^\kappa$ be a bilinear group and the order p of these two groups is is $p = p_1p_2p_3$, where p_1, p_2, p_3 are primes. Let $\mathbb{G}_{p_i}$ be the subgroup of order p_i, for $i \in \{1, 2, 3\}$ and let $\mathbb{G}_{p_ip_j}$ be the subgroup of order p_ip_j, for $i \neq j$. Observe that when we have $(u, v) \in \mathbb{G}_{p_ip_j}$ of order p_i and p_j, respectively, then $e(u, v) = 1_{\mathbb{G}_t}$. We state the assumptions very informal below:

- **Ass. 1** For given elements $g \in \mathbb{G}_{p_1}, X_3 \in \mathbb{G}_{p_3}$ and a group element T, it is hard to decide whether $T \in \mathbb{G}_{p_1p_2}$ or $T \in \mathbb{G}_{p_1}$.
- **Ass. 2** Let us assume that $g, X_1 \in \mathbb{G}_{p_1}$, $X_2, Y_2 \in \mathbb{G}_{p_2}$ and $Y_3, Z_3 \in \mathbb{G}_{p_3}$. Then it is hard to decide, given a tuple (g, X_1X_2, Z_3, Y_2Y_3) and T, if $T \in (G)$ or $T \in \mathbb{G}_{p_1p_3}$.
- **Ass. 3** Let us assume that $g \in \mathbb{G}_{p_1}$, $X_2, Y_2, Z_2 \in \mathbb{G}_{p_2}$, $X_3 \in \mathbb{G}_{p_3}$ and $\alpha, s \in \mathbb{Z}_p$. Then given the tuple $(g, g^\alpha X_2, X_3, g^s Y_2, Z_2)$ it is hard to compute $e(g, g)^{\alpha s}$.
- **Ass. 4** Let us assume that, for $t \in \mathbb{Z}_p$, the elements $g, w, g^t, X_1 \in \mathbb{G}_{p_1}$, $X_2, Y_2, Z_2 \in \mathbb{G}_{p_2}$ and $X_3, Y_3, Z_3 \in \mathbb{G}_{p_3}$ are given. Having the element $T \in \mathbb{G}$ and the tuple $(g, w, g^t, X_1X_2, X_3, Y_2Y_3)$ it is hard to decide if $T = w^t Z_3$ or $T = w^t Z_2Z_3$.
- **Ass. 5** Let us assume that $a, b, c \in \mathbb{Z}_p$, $g \in \mathbb{G}_{p_1}$, $X_2, Y_2, Z_2 \in \mathbb{G}_{p_2}$ and $X_3 \in \mathbb{G}_{p_3}$. Then given the tuple $(g, g^a, g^b, g^{ab}X_2, X_3, g^cY_2, Z_2)$ it is hard to compute $e(g, g)^{abc}$.

Now, let us come to a classic assumption, i.e., the strong RSA (sRSA) assumption [BP97]. Basically, this assumption is related to the classical RSA assumption but the exponent e could be chosen depending on c, while in the RSA problem e is chosen independently.

Definition 10 (sRSA)**.** *Let $N = pq$ be an RSA modulus of length $\kappa \in \mathbb{N}$ and p and q distinct appropriate prime numbers and c a random element in $\mathbb{Z}_N$, then for all PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that:*

$$\mathsf{Pr}\left[(m, e) \leftarrow \mathcal{A}(c, N) : m^e \equiv c \pmod{N}\right] \leq \epsilon(\kappa)$$

Furthermore, we require some hardness assumptions related to lattices.

Definition 11. *Given a uniform and random matrix $A \in \mathbb{Z}_q^{n\times m}$ for positive integers m, n and given an integer q, the* small integer solution (SIS) *is the following problem: find a nonzero integer vector $x \in \mathbb{Z}_q^m$ such that $Ax = 0 \mod q$.*

Definition 12. *Given a matrix $A \in \mathbb{Z}_q^{n\times m}$ and k short vectors $x_1, x_2, \ldots, x_k \in \mathbb{Z}^m$ such that $A \cdot x_i = 0 \mod q$ for any $i \in \{1, 2, \ldots, k\}$, the* k-small integer solution (k-SIS) *problem consists of finding another short vector $x \in \mathbb{Z}^m \backslash \mathbb{Q} - \mathrm{span}\{x_1, x_2, \ldots, x_k\}$ such that $A \cdot x = 0 \mod q$.*

A variant of the above problem, called inhomogeneous small integer solution, was introduced in [GPV08]. It consists of finding a short solution to a random inhomogeneous system.

Definition 13. *Given a uniformly random integer q, a uniformly random matrix $A \in {\mathbb{Z}_q}^{n \times m}$, a syndrome $u \in {\mathbb{Z}_q}^n$, the* inhomogeneous small integer solution (ISIS) *is the following problem: find a vector of integers $x \in \mathbb{Z}^m$ such that $Ax = u \mod q$.*

Remark 1.1. *All the above hardness assumptions can be seen in the ℓ_2 norm if we assume that there exist a real number β such that $||x|| \leq \beta$.*

1.3.3 Digital Signatures

Digital signatures are an important cryptographic tool that allow to authenticate a message in a way that everybody using the public key of the source can determine the source of the message (authenticity) and whether the source really has authenticated the message at hand (integrity). Consequently, digital signatures are often seen as the digital equivalent to handwritten signatures (while they actually provide much stronger guarantees as the analogue version). More formally, digital signatures can be defined as follows.

Definition 14 (Digital Signatures)**.** *A digital signature scheme* DSS *is a triple* (DKeyGen, DSign, DVerify) *of PPT algorithms:*

$\mathsf{DKeyGen}(1^\kappa)$**:** *The key generation algorithm that takes a security parameter $\kappa \in \mathbb{N}$ as input and outputs a secret (signing) key* sk *and a public (verification) key* pk *with associated message space $\mathcal{M}$.*[4]*.*

$\mathsf{DSign}(m, \mathsf{sk})$**:** *The (probabilistic) signing algorithm, which takes a message $m \in \mathcal{M}$ and a secret key* sk *as input, and outputs a signature σ.*

$\mathsf{DVerify}(\sigma, m, \mathsf{pk})$**:** *The deterministic verification algorithm, which takes a signature σ, a message $m \in \mathcal{M}$ and a public key* pk *as input, and outputs 1 if σ is a valid signature for m under* pk *or 0 otherwise.*

A digital signature scheme is required to be correct, i.e., for all security parameters κ, all $(\mathsf{sk}, \mathsf{pk})$ generated by DKeyGen and all $m \in \mathcal{M}$ one requires $\mathsf{DVerify}(\mathsf{DSign}(m, \mathsf{sk}), M, \mathsf{pk}) = 1$.[5]

For security one requires existential unforgeability under adaptively chosen-message attacks (EUF-CMA) [GMR88], which is defined as follows:

Definition 15 (EUF-CMA)**.** *A* DSS *is* EUF-CMA *secure, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that*

$$\mathsf{Pr}\left[\begin{array}{c}(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{DKeyGen}(1^\kappa), (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{DSign}}(\cdot,\mathsf{sk})}(\mathsf{pk}) : \\ \mathsf{DVerify}(\sigma, m^*, \mathsf{pk}) = 1 \ \wedge \ m^* \notin \mathcal{Q}^{\mathsf{DSign}}\end{array}\right] \leq \epsilon(\kappa) \ ,$$

where $\mathcal{A}$ has access to an oracle $\mathcal{O}^{\mathsf{DSign}}$ that allows to execute the DSign *algorithm and the environment keeps track of all queries to $\mathcal{O}^{\mathsf{DSign}}$ via $\mathcal{Q}^{\mathsf{DSign}}$.*

[4] We usually omit to mention the message space $\mathcal{M}$ and assume that it is implicit in the public key

[5] One may also tolerate that the DVerify algorithm fails with some negligible probability, which we however do not consider explicitly.

Hash-then-Sign. In digital signature schemes the message space is limited to some scheme (and security parameter) specific set $\mathcal{M}$. In order to extend the message space to (nearly) arbitrary message sizes one usually applies the so called hash-then-sign paradigm.

This paradigm takes an EUF-CMA secure *DSS* Σ and constructs an EUF-CMA secure *DSS* Σ' for arbitrary message spaces as follows: Use a collision resistant hash function H to map any arbitrary length message m to $\mathcal{M}$ before applying Sign of Σ.

The following theorem is straightforward to prove and can be found for instance in [Kat10].

Theorem 1.1. *If Σ is EUF-CMA secure and H is collision resistant, then Σ' is EUF-CMA secure*

For a more extensive treatment and relaxed security notions of digital signature schemes (which are not required here) we refer the reader to the excellent textbook by Katz [Kat10].

1.3.4 Public Key Encryption

Definition 16 (Public Key Encryption)**.** *A public-key encryption scheme* ENC *is a tuple of PPT algorithms* $(\mathsf{EKeyGen}, \mathsf{Enc}, \mathsf{Dec})$*:*

$\mathsf{EKeyGen}(1^\kappa)$**:** *This probabilistic algorithm takes a security parameter κ and outputs a pair of keys $(\mathsf{sk}, \mathsf{pk})$ (*pk *fixes plaintext a space $\mathcal{M}$ and a ciphertext space $\mathcal{C}$, which me do not make explicit below).*

$\mathsf{Enc}(m, \mathsf{pk})$**:** *This (probabilistic) encryption algorithm takes a message $m \in \mathcal{M}$ and a public key* pk *and outputs a ciphertext $c \in \mathcal{C}$.*

$\mathsf{Dec}(c, \mathsf{sk})$**:** *This deterministic decryption algorithm takes a ciphertext $c \in \mathcal{C}$ and a private key* sk *and outputs $m \in \mathcal{M} \cup \{\perp\}$ (where $\perp$ indicates an error).*

For correctness one requires that

$$\forall(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{EKeyGen}(1^\kappa) : \Pr\left[\mathsf{Dec}(\mathsf{Enc}(m, \mathsf{pk}), \mathsf{sk}) = m\right] = 1 - \epsilon(\kappa)$$

where the probability is required to be 1 in case of perfect correctness. Moreover, as we do only require indistinguishability under chosen-plaintext attacks (IND-CPA) security in this report we subsequently define IND-CPA and refer the reader to the literature for stronger security definitions.

Definition 17 (IND-CPA)**.** *A* ENC *scheme is called IND-CPA secure, if for all PPT adversaries $\mathcal{A}$ there is a negligible function ϵ such that:*

$$\Pr\left[\begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{EKeyGen}(1^\kappa), ((m_0, m_1), \mathsf{state}) \leftarrow \mathcal{A}(\mathsf{pk}), \\ b \leftarrow \{0,1\}, c \leftarrow \mathsf{Enc}(m_b, \mathsf{pk}), b^* \leftarrow \mathcal{A}(\mathsf{state}, c) : b^* = b\end{array}\right] \leq \frac{1}{2} + \epsilon(\kappa).$$

1.3.5 Non-Interactive Proof Systems

Definition 18 (Non-Interactive Proof System)**.** *A non-interactive proof system* Π *is a tuple of algorithms* (Gen, Proof, Verify)*, which are defined as follows:*

$\mathsf{Gen}(1^\kappa)$**:** *A PPT algorithm that takes a security parameter* κ*, and outputs a common reference string* crs*.*

$\mathsf{Proof}(\mathsf{crs}, x, w)$**:** *An algorithm that takes a common reference string* crs*, a statement* x*, and a witness* w*, and outputs a proof* π*.*

$\mathsf{Verify}(\mathsf{crs}, x, \pi)$**:** *A PPT algorithm that takes a common reference string* crs*, a statement* x*, and a proof* π*, and outputs 1 if* π *is valid and 0 otherwise.*

If Proof also runs in PPT, we are talking about a non-interactive *argument* system. Followingly, we state the required security properties (adapted from [BGI14, CKLM14, GS08, Mei09]). Let R be an NP-relation such that $(x, w) \in R$ iff w is a satisfying witness for x. Furthermore, let L_R be the corresponding NP-language of statements in R.

A non-interactive zero-knowledge proof system is required to satisfy the following security definitions:

Definition 19 (Completeness)**.** *A non-interactive proof system* (Gen, Proof, Verify) *is* complete*, if for every adversary* $\mathcal{A}$ *it holds that*

$$\Pr\Big[\mathsf{crs} \leftarrow \mathsf{Gen}(1^\kappa), (x, w) \leftarrow \mathcal{A}(\mathsf{crs}), \pi \leftarrow \mathsf{Proof}(\mathsf{crs}, x, w) : \mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 \ \wedge \ (x, w) \in R\Big] = 1.$$

Definition 20 (Soundness)**.** *A non-interactive proof system* (Gen, Proof, Verify) *is* sound*, if for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\epsilon(\cdot)$ *such that*

$$\Pr\Big[\mathsf{crs} \leftarrow \mathsf{Gen}(1^\kappa), (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) : \mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 \ \wedge \ x \notin L_R\Big] \leq \epsilon(\kappa).$$

$(\mathsf{Gen}, \mathsf{Proof}, \mathsf{Verify})$ *is perfectly sound, if* $\epsilon = 0$*.*

Definition 21 (Witness Indistinguishability)**.** *A non-interactive proof system is witness indistinguishable, if for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\epsilon(\cdot)$ *such that*

$$\Pr\Big[\mathsf{crs} \xleftarrow{R} \mathsf{Gen}(1^\kappa), b \xleftarrow{R} \{0,1\}, (x, w_0, w_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{crs}), \pi \leftarrow \mathsf{Proof}(\mathsf{crs}, x, w_i), b^* \leftarrow \mathcal{A}(\pi, \mathsf{st}) : b = b^* \ \wedge \ (x, w_0) \in R \ \wedge \ (x, w_1) \in R\Big] \leq \frac{1}{2} + \epsilon(\kappa).$$

$(\mathsf{Gen}, \mathsf{Proof}, \mathsf{Verify})$ *is perfectly witness indistinguishable, if* $\epsilon = 0$*.*

Definition 22 (Adaptive Zero-Knowledge)**.** *A non-interactive proof system is* adaptive zero-knowledge*, if there exists a simulator $S = (S_1, S_2)$ such that for every PPT adversary $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that*

$$\Big|\mathsf{Pr}\Big[\mathsf{crs} \xleftarrow{R} \mathsf{Gen}(1^\kappa) : \mathcal{A}^{\mathcal{P}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1\Big] - \mathsf{Pr}\Big[(\mathsf{crs}, \tau_s) \xleftarrow{R} S_1(1^\kappa) : \mathcal{A}^{\mathcal{S}(\mathsf{crs},\tau_s,\cdot,\cdot)}(\mathsf{crs}) = 1\Big]\Big| \leq \epsilon(\kappa),$$

where, τ_s denotes a simulation trapdoor. Thereby, $\mathcal{P}$ and $\mathcal{S}$ return $\perp$ if $(x, w) \notin R$ or $\pi \xleftarrow{R} \mathsf{Proof}(\mathsf{crs}, x, w)$ and $\pi \xleftarrow{R} S_2(\mathsf{crs}, \tau_s, x)$, respectively, otherwise. $(\mathsf{Gen}, \mathsf{Proof}, \mathsf{Verify})$ *is perfect adaptively zero-knowledge, if $\epsilon = 0$.*

We talk about zero-knowledge proofs of knowledge (ZKPoK) or zero-knowledge arguments of knowledge (ZKAoK), if they satisfy the following extractability definition.

Definition 23 (Extractability)**.** *A non-interactive proof system* $(\mathsf{Gen}, \mathsf{Proof}, \mathsf{Verify})$ *is* extractable*, if there exists an extractor $E = (E_1, E_2)$ such that (1) for every PPT adversary $\mathcal{A}$ it holds that*

$$\mathsf{Pr}\Big[\mathsf{crs} \xleftarrow{R} \mathsf{Gen}(1^\kappa) : \mathcal{A}(\mathsf{crs}) = 1\Big] \approx \mathsf{Pr}\Big[(\mathsf{crs}_e, \tau_e) \xleftarrow{R} E_1(1^\kappa) : \mathcal{A}(\mathsf{crs}_e) = 1\Big]$$

and (2) for every PPT adversary $\mathcal{A}'$ there is a negligible function $\epsilon(\cdot)$ such that

$$\mathsf{Pr}\Big[(\mathsf{crs}_e, \tau_e) \xleftarrow{R} E_1(1^\kappa), (x, \pi) \xleftarrow{R} \mathcal{A}'(\mathsf{crs}_e) : \mathsf{Verify}(\mathsf{crs}_e, x, \pi) = 1 \ \wedge \ (x, E_2(\mathsf{crs}_e, \tau_e, x, \pi)) \notin R\Big] \leq \epsilon(\kappa).$$

$(\mathsf{Gen}, \mathsf{Proof}, \mathsf{Verify})$ *is perfectly extractable, if $\epsilon = 0$ and crs_e as well as* crs *are distributed identically.*

Malleable Proofs. In the following, we are going to consider NP-relations R, which are closed with regard to a transformation $T = (T_{\mathsf{inst}}, T_{\mathsf{wit}})$ meaning that if $(x, w) \in R$, then also $(T_{\mathsf{inst}}(x), T_{\mathsf{wit}}(w)) \in R$. Then, a malleable proof system, is a non-interactive proof system defined for such a relation R and augmented by an additional algorithm ZKEval whose purpose is to transform proofs by applying T.

Definition 24 (Malleable Proof System [CKLM12])**.** *A non-interactive proof system $\Pi =$* $(\mathsf{Gen}, \mathsf{Proof}, \mathsf{Verify})$ *is* malleable *with respect to a set of transformations $\mathcal{T}$, if there exists an efficient algorithm* ZKEval*:*

$\mathsf{ZKEval}(\mathsf{crs}, T, x, \pi)$**:** *On input a CRS, a transformation $T \in \mathcal{T}$, an instance x plus a proof π such that $\mathsf{Verify}(\mathsf{crs}, x, \pi) = 1$, this algorithm outputs a proof π' for $x' = T(x)$ such that $\mathsf{Verify}(\mathsf{crs}, x', \pi') = 1$.*

We speak of *controlled malleability*, if an extractor—when given an adversarially generated proof π for some instance x—is able to extract either a witness w or a transformation $T \in \mathcal{T}$ plus a previous instance x' such that $x = T_{\mathsf{inst}}(x')$ from π:

Definition 25 (Controlled-Malleable Simulation-Sound Extractability [CKLM12])**.** *A NIZKPoK system* (Gen, Proof, Verify, ZKEval) *being malleable with respect to a set of unary transformations $\mathcal{T}$ defined on some relation R and having a simulator/extractor $SE = (SE_1, S_2, E_2)$ is* controlled-malleable simulation-sound extractable*, if for every PPT adversary $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{l}(\mathsf{crs}, \tau_s, \tau_e) \xleftarrow{R} SE_1(1^\kappa),\\ (x, \pi) \xleftarrow{R} \mathcal{A}^{S_2(\mathsf{crs}, \tau_s, \cdot)}(\mathsf{crs}, \tau_e),\\ (w, x', T) \leftarrow E_2(\mathsf{crs}, \tau_e, x, \pi), \qquad : b = 1 \wedge \mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 \wedge (x, \pi) \notin Q\\ b \leftarrow \quad (w \neq \bot \wedge (x, w) \notin R) \vee\\ \qquad\quad ((x', T) \neq (\bot, \bot)) \wedge\\ \qquad\quad (x' \notin Q_{\mathsf{inst}} \vee\\ \qquad\quad\ x \neq T_{\mathsf{inst}}(x') \vee\\ \qquad\quad\ T \notin \mathcal{T}) \vee\\ \qquad\quad (w, x', T) = (\bot, \bot, \bot))\end{array}\right] \leq \epsilon(\kappa),$$

where $Q = Q_{\mathsf{inst}} \times Q_{\mathsf{proof}}$ is a list to keep track of instances queried to S_2 and given that membership tests in $\mathcal{T}$ are efficient.

Informally, the following property demands that an adversary should be unable to distinguish transformed proofs from simulated transformed proofs.

Definition 26 (Derivation Privacy [CKLM12])**.** *A NIZKPoK system* (Gen, Proof, Verify, ZKEval) *being malleable with respect to a set of transformations $\mathcal{T}$ defined on some relation R and having a simulator $S = (S_1, S_2)$ is* derivation private*, if for every PPT adversary $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{l}(\mathsf{crs}_s, \tau_s) \xleftarrow{R} S_1(1^\kappa),\\ (\mathsf{st}, (x_i, \pi_i)_{i \in [q]}, T) \xleftarrow{R} \mathcal{A}(\mathsf{crs}_s, \tau_s),\\ \mathsf{Return}\ \bot\ \mathsf{if}\ (T \notin \mathcal{T}\ \vee\ \exists i \in [q] : \mathsf{Verify}(\mathsf{crs}_s, x_i, \pi_i) = 0),\\ \mathsf{Else\ if}\ b = 0 : \pi \xleftarrow{R} S_2(\mathsf{crs}_s, T_{\mathsf{inst}}((x_i)_{i \in [q]})), \qquad : b = b^*\\ \mathsf{Else\ if}\ b = 1 : \pi \xleftarrow{R} \mathsf{ZKEval}(\mathsf{crs}_s, T, (x_i, \pi_i)_{i \in [q]}),\\ b^* \xleftarrow{R} \mathcal{A}(\mathsf{st}, \sigma)\end{array}\right] \leq \epsilon(\kappa).$$

1.3.6 Indistinguishability Obfuscation for Random Oracles

For the construction of operational signature schemes (cf. Section 2.5) we need a combination of indistinguishability obfuscation ($i\mathcal{O}$) [BGI+12, GGH+13] and random oracles which has been introduced in [BDF+14]. We briefly sketch the idea of their approach and then provide their formal definition. Basically, the idea is to have an obfuscator for a curcuit $C^H(\cdot)$ with access to a random oracle H which is of the form $C^H(x) = C'(x, H(x))$ and the obfuscation is performed for the C' part (for which indistinguishability is required) and the random oracle is instantiated by some real code for H in any instantiation (where it has to be guaranteed that one cannot inject intermediate values into the circuit, e.g., after evaluating H).

Definition 27 (Oracle Indistinguishability Obfuscator [BDF+14])**.** *A uniform PPT machine $i\mathcal{O}$ is called an oracle indistinguishability obfuscator with respect to an oracle h for a class $\{C_\kappa\}$ of oracle circuits with upstream hashing only, if the following conditions are satisfied:*

Correctness: *For all security parameters $\kappa \in \mathbb{N}$, for all $C^H(\cdot) = C'(\cdot, H(\cdot)) \in C_\kappa$ with upstream hashing only, and for all inputs x. we have that*

$$\mathsf{Pr}[O(x, H(x)) = C^H(x) : O \leftarrow i\mathcal{O}(\kappa, C') \text{ for } C^H(\cdot) = C'(\cdot, H(\cdot))] = 1.$$

Indistinguishability: *For any (not necessarily uniform) PPT adversaries* Samp*, $\mathcal{D}$, there exists a negligible function $\epsilon(\cdot)$ such that the following holds: if $\mathsf{Pr}[C_0^H(x) = C_1^H(x) : (C_0^H, C_1^H, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\kappa)] > 1 - \epsilon(\kappa)$, then we have:*

$$\Big|\mathsf{Pr}[\mathcal{D}^H(\mathsf{aux}, i\mathcal{O}(\kappa, C_0')) = 1 : (C_0^H, C_1^H, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\kappa)]$$
$$-\mathsf{Pr}[\mathcal{D}^H(\mathsf{aux}, i\mathcal{O}(\kappa, C_1')) = 1 : (C_0^H, C_1^H, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\kappa)]\Big| \leq \epsilon(\kappa)$$

where $C_b^H(x) = C_b'(x, H(x))$ for $b \in \{0, 1\}$.

2 General Frameworks

In this section we review general framework that cover different types of functional as well as malleable/homomorphic signature schemes. In particular, we review a framework denoted functional digital signatures (cf. Section 2.1) and another denoted as policy-based signatures (cf. Section 2.2). We consider both of them to be in the class of functional signature schemes. Then, we review malleable signatures for general transformations (cf. Section 2.3) as well as P-homomorphic signatures (cf. Section 2.4), which cover different instantiations of malleable signatures for arithmetics and editing. Finally, we review operational signatures (cf. Section 2.5), which capture the aforementioned frameworks of functional signatures, P-homomorphic signatures and also various other types of other malleable signatures for editing (such as sanitizable signatures to be discussed in Section 5). Moreover, operational signatures also cover other classes of signatures such as ring [RST01] or aggregate signatures [BGLS03a].

2.1 Functional Digital Signatures

Like in a conventional signature scheme, in a *functional digital signature* (FS) scheme [BGI14] there is a signing key sk (called a *master signing key* in FS) that allows to produce a signature on any message. However, additionally there are *secondary signing keys* sk_f, which are parametrized by a function f and those restrict the signing capabilities to messages in the range of f, i.e., given any m the key sk_f only allows to produce signatures for $f(m)$.[6]

2.1.1 Formal Definitions

Definition 28 (Functional Signatures [BGI14])**.** *A functional signature (FS) scheme for a message space $\mathcal{M}$ and function family $\mathcal{F} = \{f : \mathcal{D}_f \to \mathcal{M}\}$ consists of the following polynomial time algorithms:*

$\mathsf{FS.Setup}(1^\kappa)$: *The setup algorithm takes as input the security parameter κ and outputs the master signing key* msk *and master verification key* mvk.

$\mathsf{FS.KeyGen}(\mathsf{msk}, f)$: *The key generation algorithm takes as input the master signing key* msk *and a function $f \in \mathcal{F}$ (represented as a circuit) and outputs a signing key sk_f for f.*

$\mathsf{FS.Sign}(f, \mathsf{sk}_f, m)$: *The signing algorithm takes as input the signing key sk_f for a function $f \in \mathcal{F}$ and an input $m \in \mathcal{D}_f$ and outputs $f(m)$ and a signature σ for $f(m)$.*

$\mathsf{FS.Verify}(\mathsf{mvk}, m, \sigma)$: *The verification algorithm takes as input a master verification key* mvk*, a message m and a signature σ and outputs 1 if the signature is valid or 0 otherwise.*

[6] We again note that in Section 3 we use the term functional signatures for a class of schemes that contains FS, but contains also other schemes that may not fall into this framework.

For correctness one requires that

$$\forall f \in \mathcal{F}, \forall m \in \mathcal{D}_f, (\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\kappa), \mathsf{sk}_f \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f),$$
$$(m, \sigma) \leftarrow \mathsf{FS.Sign}(f, \mathsf{sk}_f, m) : \mathsf{FS.Verify}(\mathsf{mvk}, m, \sigma) = 1.$$

Furthermore, as within conventional signature schemes one requires unforgeability under adaptive chosen message attacks. Additionally, one requires the properties *function privacy* and *succinctness*.

For unforgeability we require the following oracles, where both oracles share a dictionary indexed by tuples $(f, i) \in \mathcal{F} \times \mathbb{N}$ whose entries are signing keys $\mathsf{sk}_f^i \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$ generated throughout the security game:

$\mathcal{O}^{\mathsf{KeyGen}}(f, i)$: If there does not exists an entry for the key (f, i), then compute $\mathsf{sk}_f^i \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$, add sk_f^i for index (f, i) to the dictionary and return sk_f^i. Otherwise look up (f, i) and return the corresponding value sk_f^i.

$\mathcal{O}^{\mathsf{Sign}}(f, i, m)$: If there does not exist an entry for (f, i) in the dictionary, then run $\mathsf{sk}_f^i \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$, add sk_f^i for index (f, i) to the dictionary and return $\mathsf{FS.Sign}(f, \mathsf{sk}_f^i, m)$. Otherwise look up (f, i), take the value sk_f^i and return $\mathsf{FS.Sign}(f, \mathsf{sk}_f^i, m)$.

Let us denote by $\mathcal{Q}^{\mathsf{KeyGen}}$ and $\mathcal{Q}^{\mathsf{Sign}}$ the queries issued to the two oracles.

Definition 29 (Unforgeability:)**.** *A FS scheme is called unforgeable, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that*

$$\mathsf{Pr}\left[\begin{array}{c}(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\kappa), \mathcal{O} \leftarrow \{\mathcal{O}^{\mathsf{KeyGen}}(\cdot,\cdot), \mathcal{O}^{\mathsf{Sign}}(\cdot,\cdot,\cdot)\}, \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{mvk}) : \mathsf{FS.Verify}(\mathsf{mvk}, m^*, \sigma^*) = 1 \ \wedge \ \nexists\, m \ \ s.t. \ \ m^* = f(m) \\ \forall (f, \cdot) \in \mathcal{Q}^{\mathsf{KeyGen}} \ \wedge \ \nexists\, (f, m) \ \ s.t. \ \ m^* = f(m) \ \ \forall (f, \cdot, m) \in \mathcal{Q}^{\mathsf{Sign}}\end{array}\right] \leq \epsilon(\kappa) \ ,$$

The notion of function privacy intuitively requires that the distribution of signatures on a message generated via different signing keys to be computationally indistinguishable, even when given all the signing keys and the master signing key (modeled below by giving the randomness r for $\mathsf{FS.Setup}$ to the adversary).

Definition 30 (Function privacy:)**.** *For all PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that*

$$\mathsf{Pr}\left[\begin{array}{c}(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\kappa; r), (f_0, st) \leftarrow \mathcal{A}(\mathsf{msk}, \mathsf{mvk}, r), \\ \mathsf{sk}_{f_0} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_0), (f_1, st) \leftarrow \mathcal{A}(\mathsf{sk}_{f_0}, st), \mathsf{sk}_{f_1} \leftarrow \mathsf{FS.KeyGen}(\\ \mathsf{msk}, f_1), (m_0, m_1, st) \leftarrow \mathcal{A}(\mathsf{sk}_{f_1}, st), b \xleftarrow{R} \{0,1\}, (f_b(m_b), \sigma^*) \leftarrow \mathsf{FS.Sign}(\\ f_b, \mathsf{sk}_{f_b}, m_b), b^* \leftarrow \mathcal{A}(\sigma^*, st) : b^* = b\end{array}\right] - \frac{1}{2} \leq \epsilon(\kappa) \ ,$$

where one requires from the challenge functions that $|f_0| = |f_1|$ (where padding may be used if there is a known upper bound). Moreover for the challenge messages one requires that $|m_0| = |m_1|$ and $f_0(m_0) = f_1(m_1)$.

Definition 31 (Succinctness:). *There exists a polynomial $s(\cdot,\cdot)$ such that for every $\kappa \in \mathbb{N}$, $f \in \mathcal{F}$, $m \in \mathcal{D}_f$*

$$\Pr\left[\begin{array}{c}(\mathsf{msk},\mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\kappa), \mathsf{sk}_f \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f),\\ (f(m),\sigma) \leftarrow \mathsf{FS.Sign}(f,\mathsf{sk}_f,m) : |\sigma| \leq s(\kappa,|f(m)|).\end{array}\right] = 1\ .$$

In particular, the signature size is independent of the size $|m|$ of the input to the function, and of the size $|f|$ of a description of the function f.

2.1.2 Constructions

Subsequently, we briefly discuss the three constructions presented in [BGI14]. The first relies on EUF-CMA secure conventional signature schemes but does neither achieve function privacy nor succintness. However, it can be instantiated based on standard assumptions, i.e., one-way functions [Rom90].

The second construction builds upon the first one, but additionally requires a zero-knowledge succinct non-interactive argument of knowledge (SNARK) system [BCCT13] in order to achieve function privacy and succinctness. This succinctness, however, comes at the cost of non-falsifiable assumptions [GW11].

Finally, the third construction drops the succinctness requirement but still achieves function privacy. This is achieved using a non-interactive zero-knowledge arguments of knowledge (NIZK-AoK) system instead of SNARKs and thus can again be based on standard assumptions.

OWF-based Construction. Below we present an OWF-based construction which we denote by FS1. For this construction let $(\mathsf{DKeyGen}, \mathsf{DSign}, \mathsf{DVerify})$ be an EUF-CMA secure conventional signature scheme (which can be constructed from any OWF [Rom90]).

$\mathsf{FS1.Setup}(1^\kappa)$: Run $(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{DKeyGen}(1^\kappa)$ and output the master signing key $\mathsf{msk} \leftarrow \mathsf{sk}$ and master verification key $\mathsf{mvk} \leftarrow \mathsf{pk}$.

$\mathsf{FS1.KeyGen}(\mathsf{msk}, f)$: Run $(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{DKeyGen}(1^\kappa)$, compute $\sigma_{\mathsf{pk}} \leftarrow \mathsf{DSign}(f||\mathsf{pk},\mathsf{msk})$, assemble a certificate $c \leftarrow (f,\mathsf{pk},\sigma_{\mathsf{pk}})$ and return $\mathsf{sk}_f \leftarrow (\mathsf{sk},c)$.

$\mathsf{FS1.Sign}(f,\mathsf{sk}_f,m)$: Parse sk_f as (sk,c), compute $\sigma_m \leftarrow \mathsf{DSign}(\mathsf{m},\mathsf{sk})$ and output $(f(m),\sigma)$ with $\sigma \leftarrow (m,c,\sigma_m)$.

$\mathsf{FS1.Verify}(\mathsf{mvk},m',\sigma)$: Parse σ as $(m, c=(f,\mathsf{pk},\sigma_{\mathsf{pk}}),\sigma_m)$ and check whether

1. $m' = f(m)$,
2. $\mathsf{DVerify}(\sigma_m, m, \mathsf{pk}) = 1$,
3. $\mathsf{DVerify}(\sigma_{\mathsf{pk}}, \mathsf{pk}||f, \mathsf{mvk}) = 1$, and

return 1 if so and 0 otherwise.

Theorem 1 ([BGI14]). *If* $(\mathsf{DKeyGen}, \mathsf{DSign}, \mathsf{DVerify})$ *is EUF-CMA secure, then* FS1 *is an unforgeable functional signature scheme.*

SNARK-based Construction. Below we present a SNARK-based construction which we denote by FS2. Therefore, let FS1 be an unforgeable FS scheme (which is not necessarily function private nor succinct). Moreover, let $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify}, \mathcal{S} = (\mathcal{S}^{\mathsf{crs}}, \mathcal{S}^{\mathsf{Proof}}), \mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2))$ be an efficient adaptive zero-knowledge SNARK system for the NP language L defined as

$$L = \{(m, \mathsf{mvk}) \mid \exists \sigma \;\; s.t. \;\; \mathsf{FS1.Verify}(\mathsf{mvk}, m, \sigma) = 1\}.$$

Now, we are ready to present the construction.

$\mathsf{FS2.Setup}(1^\kappa)$: Run $(\mathsf{mvk}', \mathsf{msk}') \leftarrow \mathsf{FS1.KeyGen}(1^\kappa)$, $\mathsf{crs} \leftarrow \Pi.\mathsf{Gen}(1^\kappa)$ and output the master signing key $\mathsf{msk} \leftarrow \mathsf{msk}'$ and master verification key $\mathsf{mvk} \leftarrow (\mathsf{msk}', \mathsf{crs})$.

$\mathsf{FS2.KeyGen}(\mathsf{msk}, f)$: Return $\mathsf{sk}_f \leftarrow \mathsf{FS1.KeyGen}(\mathsf{msk}, f)$.

$\mathsf{FS2.Sign}(f, \mathsf{sk}_f, m)$: Compute $(f(m), \sigma') \leftarrow \mathsf{FS1.Sign}(f, \mathsf{sk}_f, m)$. Additionally, generate $\pi \leftarrow \Pi.\mathsf{Proof}((f(m), \mathsf{mvk}'), \sigma', \mathsf{crs})$ and return $(f(m), \pi)$.

$\mathsf{FS2.Verify}(\mathsf{mvk}, m', \sigma)$: Output $\Pi.\mathsf{Verify}(\mathsf{crs}, m', \sigma)$.

Theorem 2 ([BGI14])**.** *If there exists an unforgeable (but not necessarily succinct or function-private) functional signature scheme* FS1 *supporting class* $\mathcal{F}$ *of polynomial-sized circuits and let* Π *be an adaptive zero-knowledge SNARK system for NP, then there exists succinct, function-private functional signatures for* $\mathcal{F}$*.*

NIZK-based Construction. In the NIZK-based construction one simply replaces the efficient adaptive zero-knowledge SNARK system Π of the construction FS2 with an non-interactive zero-knowledge argument of knowledge (NIZKAoK) system Π'.

Theorem 3 ([BGI14])**.** *If* (DKeyGen, DSign, DVerify) *is EUF-CMA secure and* Π' *is a NIZKAoK system, then* FS3 *is an unforgeable and function-private functional signature scheme.*

2.1.3 Applications

Functional signatures have two very interesting applications, which we are going to briefly sketch below.

Delegation of Signing Rights. Functional signatures can be seen as a generalization of proxy-signature schemes (cf. Section 3.1). Loosely speaking proxy-signature schemes allow a party (the delegator) to delegate signing rights (for a well defined set of messages) to another party (the delegatee). The delegatee can then produce signatures on behalf of the delegator (for allowed messages). For instance, as discussed in [BGI14], with a functional signature scheme, the holder of a master signing key msk could delegate a signing key sk_f with

$$f(m) = \begin{cases} m \text{ if } P(m) = 1 \\ \perp \text{ otherwise.} \end{cases}$$

and $P(\cdot)$ being a predicate such as "signed by delegatee x" to a delegatee. The predicate could also be more complex and check if some certain substring is present or not or if it satisfies a certain policy. If the FS scheme is function-private, then this policy can also be hidden, yielding additional privacy guarantees.

Function privacy (to a more limited degree) has also been investigated in context of proxy-signatures and schemes that are function private are there called warrant-hiding proxy-signature schemes [HS13b]. However, in contrast to functional signature schemes where the function f could be an arbitrary circuit, the aforementioned proxy-signature schemes only consider functions as being an enumeration of it's range. Thus only functions with compact enumerations of it's range can be supported. Nevertheless, known constructions (cf. [HS13b]) achieve succinctness.

Verifiable Computations. Functional signatures can be straightforwardly used to construct verifiable computation schemes with efficient verification as discussed in [BGI14]. In such schemes a client outsources the computation of some function f for some client-chosen input m to some other party, who—along with the result $f(m)$—delivers some additional information π that allows to efficiently check the correctness of the result $f(m)$. Here, the client can give the other party a key $\mathsf{sk}_{f'}$ for a function $f'(x) := (f(x)||x)$. Then, to prove that for a result y returned to the client it indeed holds that $y = f(x)$, the prover issues a signature for $y||x$ which the prover can only obtain if $y||x$ is in the range of f' and thus $y = f(x)$.

Finally, we note that Papamanthou et al. [PST13] have defined a concept denoted *signatures of correct computation*, which is similar to the framework of functional signatures. However, their approach is tailored towards the application of verifiably outsourcing computations.

2.2 Policy-Based Signatures

Policy-based signatures (PBS) [BF14] allow to restrict the message space a signer is allowed to sign via so called *policies*. More precisely, a trusted authority issues a secret signing key sk_p to a signer. This key corresponds to a policy $p \in \{0,1\}^*$ and allows to issue signatures σ on messages $m \in \{0,1\}^*$ if (p, m) is contained in a language $L_{\mathsf{PC}} \subseteq \{0,1\}^* \times \{0,1\}^*$. A tuple (p, m) is contained in this language if there exists a witness w attesting that m satisfies p, i.e., $\mathsf{PC}((m, p), w) = 1$, where $\mathsf{PC} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ is an NP-relation. Signatures are then verified using the authority's public parameters pp. We note that the policy in PBS can be seen as a function f in the framework of functional digital signatures (cf. Section 2.1).

2.2.1 Formal Definitions

Subsequently, we recall the model of PBS.

Definition 32 (PBS Scheme)**.** *A* PBS *scheme is a tuple* (Setup, KeyGen, Sign, Verify) *of efficient algorithms, which are defined as follows:*

$\mathsf{PBS.Setup}(1^\kappa)$**:** *This algorithm takes a security parameter* κ *as input and outputs the public parameters* pp *and a master secret key* msk*.*

$\mathsf{PBS.KeyGen}(\mathsf{pp}, \mathsf{msk}, p)$**:** *This algorithm takes public parameters* pp*, a master secret key* msk *and a policy* p *as input and outputs a signing key* sk_p *for* p*.*

$\mathsf{PBS.Sign}(\mathsf{pp}, \mathsf{sk}_p, m, w)$**:** *This algorithm takes public parameters* pp*, a signing key* sk_p*, a message* m *and a witness* w *as input and outputs a signature* σ*.*

$\mathsf{PBS.Verify}(\mathsf{pp}, m, \sigma)$**:** *This algorithm takes public parameters* pp*, a message* m *and a signature* σ *as input and outputs a bit* b*.*

For security, PBS are required to be correct, unforgeable and indistinguishable. In addition, PBS may provide simulatability and extractability, which can be seen as even stronger security requirements. That is, any PBS that is simulatable and extractable is also unforgeable and indistinguishable [BF14, Theorem 1]. Below, we recall the respective security definitions (adapted to our notation).

Definition 33 (Unforgeability)**.** *A* PBS *scheme is called unforgeable, if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\epsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{c}(\mathsf{pp},\mathsf{msk}) \xleftarrow{R} \mathsf{PBS.Setup}(1^\kappa), \mathcal{O} \leftarrow \{\mathcal{O}^{\mathsf{KeyGen}}(\mathsf{pp},\mathsf{msk},\cdot,\cdot), \mathcal{O}^{\mathsf{RevealSK}}(\cdot),\\ \mathcal{O}^{\mathsf{Sign}}(\mathsf{pp},\cdot,\cdot,\cdot)\}, (m,\sigma) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}) : \mathsf{PBS.Verify}(\mathsf{pp},m,\sigma) = 1 \ \wedge \\ \nexists\, i : (m \in \mathcal{Q}^{\mathsf{Sign}}[i] \ \vee \ (m, \mathcal{Q}^{\mathsf{RevealSK}}[i][1]) \in L_{\mathsf{PC}})\end{array}\right] \leq \epsilon(\kappa),$$

where the environment maintains the lists $\mathcal{Q}^{\mathsf{KeyGen}}, \mathcal{Q}^{\mathsf{RevealSK}}$ *and* $\mathcal{Q}^{\mathsf{Sign}}$ *to keep track of the oracle queries and the oracles are defined as follows:*

$\mathcal{O}^{\mathsf{KeyGen}}(\mathsf{pp}, \mathsf{msk}, i, p)$ *takes public parameters* pp*, a master secret key* msk*, an index* i *and a policy* p *as input. If* $\mathcal{Q}^{\mathsf{KeyGen}}[i] \neq \emptyset$*, it returns* $\perp$*. Otherwise, it runs* $\mathsf{sk}_p \xleftarrow{R} \mathsf{PBS.KeyGen}(\mathsf{pp}, \mathsf{msk}, p)$ *and sets* $\mathcal{Q}^{\mathsf{KeyGen}}[i] \leftarrow (p, \mathsf{sk}_p)$*.*

$\mathcal{O}^{\mathsf{RevealSK}}(i)$ *takes an index* i *as input and returns* $\perp$ *if* $\mathcal{Q}^{\mathsf{KeyGen}}[i] = \emptyset$*. Otherwise it obtains* $(p, \mathsf{sk}_p) \leftarrow \mathcal{Q}^{\mathsf{KeyGen}}[i]$*, sets* $\mathcal{Q}^{\mathsf{RevealSK}}[i] \leftarrow (p, \mathsf{sk}_p)$ *and returns* sk_p*.*

$\mathcal{O}^{\mathsf{Sign}}(\mathsf{pp}, i, m, w)$ *takes public parameters* pp*, an index* i*, a message* m *and a witness* w*. If* $\mathcal{Q}^{\mathsf{KeyGen}}[i] = \emptyset$ *or* $\mathcal{Q}^{\mathsf{RevealSK}}[i] \neq \emptyset$ *it returns* $\perp$*. Otherwise, it sets* $\mathcal{Q}^{\mathsf{Sign}}[i] \leftarrow \mathcal{Q}^{\mathsf{Sign}}[i] \cup \{m\}$ *and returns* $\mathsf{PBS.Sign}(\mathsf{pp}, \mathcal{Q}^{\mathsf{KeyGen}}[i][2], m, w)$*.*

Definition 34 (Indistinguishability)**.** *A* PBS *scheme is called indistinguishable, if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\epsilon(\cdot)$ *such that*

$$\Pr\Big[(\mathsf{pp},\mathsf{msk}) \xleftarrow{R} \mathsf{PBS.Setup}(1^\kappa), b \xleftarrow{R} \{0,1\}, \\ b^* \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{LoR}}(\mathsf{pp},\mathsf{msk},\cdot,\cdot,\cdot,\cdot,\cdot,b)}(\mathsf{pp},\mathsf{msk}) : b = b^*\Big] \leq \frac{1}{2} + \epsilon(\kappa),$$

where $\mathcal{O}^{\mathsf{LoR}}$ *is defined as follows:*

$\mathcal{O}^{\mathsf{LoR}}(\mathsf{pp}, \mathsf{msk}, p_0, p_1, m, w_0, w_1, b)$: *This oracle takes public parameters* pp *a master secret key* msk*, a bit* b*, two policies* p_0, p_1*, a message* m*, two witnesses* w_0, w_1 *and a bit* b *as input. If* $\mathsf{PC}((p_0, m), w_0) \neq 1$ *or* $\mathsf{PC}((p_1, m), w_1) \neq 1$ *it returns* $\perp$*. Otherwise, it computes* $\mathsf{sk}_{p_0} \xleftarrow{R} \mathsf{PBS.KeyGen}(\mathsf{pp}, \mathsf{msk}, p_0)$, $\mathsf{sk}_{p_1} \xleftarrow{R} \mathsf{PBS.KeyGen}(\mathsf{pp}, \mathsf{msk}, p_1)$, $\sigma_b \xleftarrow{R} \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_b, m, w_b)$ *and returns* $(\sigma_b, \mathsf{sk}_{p_0}, \mathsf{sk}_{p_1})$.

Definition 35 (Simulatability)**.** *A* PBS *is called simulatable if there exists a PPT simulator* $\{\mathsf{SimSetup}, \mathsf{SimKeyGen}, \mathsf{SimSign}\}$*, such that for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\epsilon(\cdot)$ *such that*

$$\left| \Pr\left[\begin{array}{c} (\mathsf{pp}, \mathsf{msk}) \xleftarrow{R} \mathsf{Setup}(1^\kappa), \\ \mathcal{O} \leftarrow \{\mathcal{O}^{\mathsf{Key}}(\mathsf{pp}, \mathsf{msk}, \cdot, \cdot), \\ \mathcal{O}^{\mathsf{Sign}}(\mathsf{pp}, \cdot, \cdot, \cdot)\} : \\ \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{msk}) = 1 \end{array} \right] - \Pr\left[\begin{array}{c} (\mathsf{pp}, \mathsf{msk}, \tau_s) \xleftarrow{R} \mathsf{SimSetup}(1^\kappa), \\ \mathcal{O} \leftarrow \{\mathcal{O}^{\mathsf{SimKey}}(\mathsf{pp}, \tau_s, \cdot, \cdot), \\ \mathcal{O}^{\mathsf{SimSign}}(\mathsf{pp}, \tau_s, \cdot, \cdot, \cdot)\} : \\ \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{msk}) = 1 \end{array} \right] \right| \leq \epsilon(\kappa)$$

where $\mathsf{SimSetup}$ *additionally outputs a simulation trapdoor* τ_s*. Furthermore, the environment internally maintains the lists* $\mathcal{Q}^{\mathsf{Key}}, \mathcal{Q}^{\mathsf{SimKey}}, \mathcal{Q}^{\mathsf{Sign}}$ *and provides oracle access to* $\mathcal{O}^{\mathsf{Key}}, \mathcal{O}^{\mathsf{SimKey}}, \mathcal{O}^{\mathsf{Sign}}$, $\mathcal{O}^{\mathsf{SimSign}}$*, which are defined as follows:*

$\mathcal{O}^{\mathsf{Key}}(\mathsf{pp}, \mathsf{msk}, i, p)$ *takes public parameters* pp*, a master secret key* msk*, an index* i *and a policy* p *as input. If* $\mathcal{Q}^{\mathsf{KeyGen}}[i] \neq \emptyset$*, it returns* $\perp$*. Otherwise, it runs* $\mathsf{sk}_p \xleftarrow{R} \mathsf{PBS.KeyGen}(\mathsf{pp}, \mathsf{msk}, p)$*, sets* $\mathcal{Q}^{\mathsf{KeyGen}}[i] \leftarrow (p, \mathsf{sk}_p)$ *and returns* sk_p*.*

$\mathcal{O}^{\mathsf{SimKey}}(\mathsf{pp}, \tau_s, i, p)$ *takes public parameters* pp*, a simulation trapdoor* τ_s*, an index* i *and a policy* p *as input. If* $\mathcal{Q}^{\mathsf{SimKeyGen}}[i] \neq \emptyset$*, it returns* $\perp$*. Otherwise, it runs* $\mathsf{sk}_p \xleftarrow{R} \mathsf{SimKeyGen}(\mathsf{pp}, \tau_s, p)$*, sets* $\mathcal{Q}^{\mathsf{SimKeyGen}}[i] \leftarrow p$ *and returns* sk_p*.*

$\mathcal{O}^{\mathsf{Sign}}(\mathsf{pp}, i, m, w)$ *takes public parameters* pp*, an index* i*, a message* m *and a witness* w*. If* $\mathcal{Q}^{\mathsf{KeyGen}}[i] = \emptyset$ *it returns* $\perp$*. Otherwise, it returns* $\mathsf{PBS.Sign}(\mathsf{pp}, \mathcal{Q}^{\mathsf{KeyGen}}[i][2], m, w)$*.*

$\mathcal{O}^{\mathsf{SimSign}}(\mathsf{pp}, \tau_s, i, m, w)$ *takes public parameters* pp*, a simulation trapdoor* τ_s*, an index* i*, a message* m *and a witness* w*. If* $\mathcal{Q}^{\mathsf{SimKeyGen}}[i] = \emptyset$ *or* $\mathsf{PC}((\mathcal{Q}^{\mathsf{SimKeyGen}}[i], m), w) \neq 1$ *it returns* $\perp$*. Otherwise, it returns* $\mathsf{SimSign}(\mathsf{pp}, \tau_s, m)$*.*

Definition 36 (Extractability)**.** *A* PBS *is called extractable if there exists a PPT simulator* $\{\mathsf{SimSetup}, \mathsf{SimKeyGen}, \mathsf{SimSign}\}$ *and a PPT extraction algorithm* Extr*, such that for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\epsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{c} (\mathsf{pp}, \mathsf{msk}, \tau_s) \xleftarrow{R} \mathsf{SimSetup}(1^\kappa), \mathcal{O} \leftarrow \{\mathcal{O}^{\mathsf{SimKey}}(\mathsf{pp}, \tau_s, \cdot), \mathcal{O}^{\mathsf{SimSign}}(\mathsf{pp}, \tau_s, \cdot)\}, \\ (m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}), (p, w) \leftarrow \mathsf{Extr}(\tau_s, m, \sigma) : \mathsf{Verify}(\mathsf{pp}, m, \sigma) = 1 \wedge \\ (m, \sigma) \notin \mathcal{Q}^{\mathsf{SimSign}} \ \wedge \ (p \notin \mathcal{Q}^{\mathsf{SimKey}} \ \vee \ \mathsf{PC}((p, m), w) \neq 1) \end{array} \right] \leq \epsilon(\kappa),$$

where the environment maintains the lists $\mathcal{Q}^{\mathsf{SimKey}}$ *and* $\mathcal{Q}^{\mathsf{SimSign}}$ *to keep track of the queries to* $\mathcal{O}^{\mathsf{SimKey}}$ *and* $\mathcal{O}^{\mathsf{SimSign}}$*, which are defined as follows:*

$\mathcal{O}^{\mathsf{SimKey}}(\mathsf{pp}, \tau_s, p)$ *takes public parameters* pp*, a simulation trapdoor* τ_s *and a policy* p *as input. It sets* $\mathcal{Q}^{\mathsf{SimKey}} \leftarrow \mathcal{Q}^{\mathsf{SimKey}} \cup \{p\}$*, and returns* $\mathsf{SimKeyGen}(\mathsf{pp}, \tau_s, p)$*.*

$\mathcal{O}^{\mathsf{SimSign}}(\mathsf{pp}, \tau_s, m)$ *takes public parameters* pp*, a simulation trapdoor* τ_s *and a message* m *as input. It computes* $\sigma \xleftarrow{R} \mathsf{SimSign}(\mathsf{pp}, \tau_s, m)$*, sets* $\mathcal{Q}^{\mathsf{SimSign}} \leftarrow \mathcal{Q}^{\mathsf{SimSign}} \cup \{(m, \sigma)\}$ *and returns* σ*.*

2.2.2 Constructions

Subsequently, we introduce the generic construction of PBS from [BF14]. It builds upon a NIZK proof system $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify})$, an EUF-CMA secure signature scheme $\mathsf{DSS} = (\mathsf{DKeyGen}, \mathsf{DSign}, \mathsf{DVerify})$, a strongly unforgeable one-time signature scheme $\mathsf{OTS} = (\mathsf{OKeyGen}, \mathsf{OSign}, \mathsf{OVerify})$ and an IND-CPA secure encryption scheme $\mathsf{ENC} = (\mathsf{EKeyGen}, \mathsf{Enc}, \mathsf{Dec})$. The NIZK proof system is used to prove knowledge of a witness for the following relation:

$$\begin{aligned}&((\mathsf{pk}_e, \mathsf{mvk}, C_p, C_s, C_w, \mathsf{pk}_{ots}, m), (p, s, w, \rho_p, \rho_s, \rho_w)) \in R \\ &\quad \Leftrightarrow C_p \leftarrow \mathsf{Enc}(p, \mathsf{pk}_e; \rho_p) \ \wedge \ C_s \leftarrow \mathsf{Enc}(s, \mathsf{pk}_e; \rho_s) \ \wedge \ C_w \leftarrow \mathsf{Enc}(w, \mathsf{pk}_e; \rho_w) \wedge \\ &\qquad [\mathsf{DVerify}(s, 1||p, \mathsf{mvk}) = 1 \ \wedge \ \mathsf{PC}((p, m), w) = 1 \ \vee \ \mathsf{DVerify}(s, 0||\mathsf{ovk}, \mathsf{mvk}) = 1].\end{aligned}$$

Now, we are ready to introduce the construction:

$\mathsf{PBS.Setup}(1^\kappa)$: This algorithm runs $\mathsf{crs} \xleftarrow{R} \mathsf{Gen}(1^\kappa)$, $(\mathsf{sk}_e, \mathsf{pk}_e) \xleftarrow{R} \mathsf{EKeyGen}(1^\kappa)$, $(\mathsf{msk}, \mathsf{mvk}) \xleftarrow{R} \mathsf{DKeyGen}(1^\kappa)$ and returns $\mathsf{pp} \leftarrow (\kappa, \mathsf{crs}, \mathsf{pk}_e, \mathsf{mvk})$ and msk.

$\mathsf{PBS.KeyGen}(\mathsf{pp}, \mathsf{msk}, p)$: This algorithm runs $s \xleftarrow{R} \mathsf{DSign}(1||p, \mathsf{msk})$ and returns $\mathsf{sk}_p \leftarrow (p, s)$.

$\mathsf{PBS.Sign}(\mathsf{pp}, \mathsf{sk}_p, m, w)$: If $\mathsf{PC}((p, m), w) \neq 1$, this algorithm returns $\perp$. Otherwise it runs $(\mathsf{sk}_{ots}, \mathsf{pk}_{ots}) \xleftarrow{R} \mathsf{OKeyGen}(1^\kappa)$, chooses $\rho_p, \rho_s, \rho_w \xleftarrow{R} \{0,1\}^\kappa$ and computes $C_p \leftarrow \mathsf{Enc}(p, \mathsf{pk}_e; \rho_p)$, $C_s \leftarrow \mathsf{Enc}(s, \mathsf{pk}_e; \rho_s)$, $C_w \leftarrow \mathsf{Enc}(w, \mathsf{pk}_e; \rho_w)$. Furthermore it computes $\pi \xleftarrow{R} \Pi.\mathsf{Prove}(\mathsf{crs}, (\mathsf{pk}_e, \mathsf{mvk}, C_p, C_s, C_w, \mathsf{pk}_{ots}, m), (p, s, w, \rho_p, \rho_s, \rho_w))$, $\tau \xleftarrow{R} \mathsf{OSign}(m||C_p||C_s||C_w||\pi, \mathsf{sk}_{ots})$ and returns $\sigma \leftarrow (\mathsf{pk}_{ots}, C_p, C_s, C_w, \pi, \tau)$.

$\mathsf{PBS.Verify}(\mathsf{pp}, m, \sigma)$: If $\Pi.\mathsf{Verify}(\mathsf{crs}, (\mathsf{pk}_e, \mathsf{mvk}, C_p, C_s, C_w, \mathsf{pk}_{ots}, m), \pi) = 1$ and $\mathsf{OVerify}(\tau, m||C_p||C_s||C_w||\pi, \mathsf{pk}_{ots}) = 1$ this algorithm returns 1 and 0 otherwise.

Subsequently, we recall how the simulator is implemented.

$\mathsf{SimSetup}(1^\kappa)$: This algorithm runs $\mathsf{crs} \xleftarrow{R} \mathsf{Gen}(1^\kappa)$, $(\mathsf{sk}_e, \mathsf{pk}_e) \xleftarrow{R} \mathsf{EKeyGen}(1^\kappa)$, $(\mathsf{msk}, \mathsf{mvk}) \xleftarrow{R} \mathsf{DKeyGen}(1^\kappa)$ and returns $\mathsf{pp} \leftarrow (\kappa, \mathsf{crs}, \mathsf{pk}_e, \mathsf{mvk})$, msk and $\tau_s \leftarrow (\mathsf{msk}, \mathsf{sk}_e)$.

$\mathsf{SimKeyGen}(\mathsf{pp}, \mathsf{msk}, p)$: This algorithm runs $s \xleftarrow{R} \mathsf{DSign}(1||p, \mathsf{msk})$ and returns $\mathsf{sk}_p \leftarrow (p, s)$.

$\mathsf{SimSign}(\mathsf{pp}, \mathsf{tk}, m, w)$: This algorithm computes $s \xleftarrow{R} \mathsf{DSign}(0||\mathsf{pk}_{ots}, \mathsf{msk})$ runs $(\mathsf{sk}_{ots}, \mathsf{pk}_{ots}) \xleftarrow{R} \mathsf{OKeyGen}(1^\kappa)$, chooses $\rho_p, \rho_s, \rho_w \xleftarrow{R} \{0,1\}^\kappa$ and computes $C_p \leftarrow \mathsf{Enc}(0, \mathsf{pk}_e; \rho_p)$, $C_s \leftarrow \mathsf{Enc}(s, \mathsf{pk}_e; \rho_s)$, $C_w \leftarrow \mathsf{Enc}(0, \mathsf{pk}_e; \rho_w)$. Furthermore it computes $\pi \xleftarrow{R} \Pi.\mathsf{Prove}(\mathsf{crs}, (\mathsf{pk}_e, \mathsf{mvk}, C_p, C_s, C_w, \mathsf{pk}_{ots}, m), (0, s, 0, \rho_p, \rho_s, \rho_w))$, $\tau \xleftarrow{R} \mathsf{OSign}(m||C_p||C_s||C_w||\pi, \mathsf{sk}_{ots})$ and returns $\sigma \leftarrow (\mathsf{pk}_{ots}, C_p, C_s, C_w, \pi, \tau)$.

$\mathsf{Extr}(\mathsf{pp}, m, \sigma)$: This algorithm computes $p \leftarrow \mathsf{Dec}(C_p, \mathsf{sk}_e)$, $w \leftarrow \mathsf{Dec}(C_w, \mathsf{sk}_e)$ and returns (p, w).

Theorem 4 ([BF14, Theorem 4]). *The construction above is simulatable and extractable (and thus also unforgeable and indistinguishable).*

The authors also proposed an *efficient* instantiation using a structure-preserving signature scheme for vectors of group elements (cf. [AFG+10]), a one-time signature scheme, and Groth-Sahai (GS) proofs [GS08]. For a detailed discussion of this instantiation, we refer the reader to [BF14]. We note that one does not need the encryption scheme when using GS proofs, since they directly allow to extract the respective witnesses.

In addition to the construction presented above, [BF14] also introduced a construction based on a single signature scheme and simulation-extractable (SE) NIZK proofs [Gro06]. This yields a simpler construction at the cost of a more complex underlying primitive (SE-NIZK). Since the basic principles of the scheme are quite similar to the construction above, we refer the reader to [BF14] for further details about the construction.

2.2.3 Applications

Likewise to the framework of functional digital signatures discussed before, PBS are a rather generic framework and thus can be used in various applications. Among others, [BF14] show that PBS imply group signatures [BMW03], attribute-based signatures [MPR11] and signatures of knowledge [CL06].

In our context, we are particularly interested in the delegation of signing rights to third parties, which can easily be achieved using PBS. In addition, [BF14] provides an extension to model hierarchies by allowing parties to further delegate a subset of their signing rights to subsequent parties. Due to the generality of the relations that can be covered by the policies, PBS may also be applicable to verifiable computations as already discussed in Section 2.1.3.

2.3 Malleable Signatures for General Transformations

Malleable signatures (cf. Section 4 and Section 5) are signature schemes, which allow controlled transformations (e.g. arithmetical ones) on message-signature pairs and, thereby, uphold the validity of the resulting message-signature pair. One important aspect of such schemes, is a property called *context-hiding*, which guarantees that derived signatures are indistinguishable from freshly generated signatures on the same message. In a nutshell, a signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is malleable if it features an additional algorithm $\mathsf{SigEval}$ that—when given (among others) message-signature pairs $(\vec{m}, \vec{\sigma})$ and an admissible transformation T—allows for obtaining a signature σ' on $m' = T(\vec{m})$. We note that transformations are very general and can implement policies such as for instance used in policy-based signatures or genereal functions[7].

In [CKLM14], Chase et. al give new definitions for malleable signature schemes having general transformations and present according security definitions. The authors do not only generalize the concepts of previous works [ABC+12, ALP12] (i.e., the framework of P-homomorphic signatures to be discussed in Section 2.4), but also strengthen previous security definitions. Additionally, they give a generic construction for malleable signatures allowing unary transfor-

[7] Actually, the authors show that their framework implies several other concepts such as policy-based signatures [BF14] (cf. Section 2.2) as well as delegatable functional signatures [BMS13] (cf. Section 2.5)

mations from controlled-malleable non-interactive zero-knowledge proofs (cm-NIZK) and show how to build delegatable anonymous credentials from it.

2.3.1 Formal Definitions

At first, we will restate the malleability definition from [CKLM14].

Definition 37 (Malleable Signature Scheme)**.** *A signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is* malleable *with regard to a set of allowed transformations* $\mathcal{T}$ *that is closed under composition if it has an additional efficient algorithm* $\mathsf{SigEval}$*:*

$\mathsf{SigEval}(\mathsf{pk}, T, \vec{m}, \vec{\sigma})$**:** *This algorithm takes as input a public key* pk*, a transformation* T *and a list of valid message-signature pairs* $\vec{m}, \vec{\sigma}$ *and outputs a signature* σ' *on* $m' = T(\vec{m})$*.*

A malleable signature scheme is required to be *unforgeable* and *context-hiding*. In the following, we will give these definitions in the context of simulatable malleable signature schemes. In doing so, we start by giving the definition of a simulatable signature scheme, which differs from the ordinary definition (cf. Section 1.3.3) in introducing a setup algorithm Gen responsible for setting up a common-reference string (CRS) crs [8] and an additional algorithm $\mathsf{KeyCheck}$:

Definition 38. *A signature scheme* $(\mathsf{Gen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is* simulatable *if there exists an efficient algorithm* $\mathsf{KeyCheck}$

$\mathsf{KeyCheck}(\mathsf{crs}, \mathsf{pk}, \mathsf{sk})$**:** *This algorithm takes as input a common-reference string* crs*, a public key* pk *and a secret key* sk *and outputs a bit* b *indicating whether* $(\mathsf{pk}, \mathsf{sk})$ *is a valid key pair under* crs*.*

and an efficient simulator $(\mathsf{SimGen}, \mathsf{SimSign})$ *such that the CRS in* $(\mathsf{crs}, \tau_s) \xleftarrow{R} \mathsf{SimGen}(1^\kappa)$ *as well as the signatures output by* $\mathsf{SimSign}$ *when given the trapdoor* τ_s *are indistinguishable from their honestly computed counterparts. This is subsumed as follows:*

$$\mathsf{Pr}[\mathsf{crs} \xleftarrow{R} \mathsf{Gen}(1^\kappa) : \mathcal{A}^{S(\mathsf{crs},\cdot,\cdot,\cdot)}(\mathsf{crs}) = 1] \approx \mathsf{Pr}[(\mathsf{crs}, \tau_s) \xleftarrow{R} \mathsf{SimGen}(1^\kappa)] : \mathcal{A}^{S'(\mathsf{crs},\tau_s,\cdot,\cdot,\cdot)}(\mathsf{crs}) = 1],$$

where S *and* S' *are oracles returning* $\perp$ *if* $\mathsf{KeyCheck}(\mathsf{crs}, \mathsf{pk}, \mathsf{sk}) = 0$ *or the result of* $\mathsf{Sign}(\mathsf{crs}, \mathsf{sk}, \mathsf{m})$ *and* $\mathsf{SimSign}(\mathsf{crs}, \tau_\mathsf{s}, \mathsf{pk}, \mathsf{m})$*, respectively, otherwise.*

Now, we are ready to detail the security properties *simulation unforgeability* and *simulation context hiding* from [CKLM14]. The intuition is as follows. In case of simulation unforgeability, the adversary should be unable to output a signature σ^* on some $m^* = T(\vec{m})$ without *knowing* T and appropriate message-signature pairs $(\vec{m}, \vec{\sigma})$, which is covered by extractability. In case of context-hiding, the idea is (as already mentioned above) that derived signatures should be indistinguishable from fresh signatures on the same message—even in front of adversarially generated keys (cf. [FHS15] for an alternative definition).

The formal definitions are as follows:

[8] All involved algorithms additionally take crs as input.

Definition 39 (Simulation Unforgeability)**.** *A simulatable signature scheme* (Gen, KeyGen, Sign, Verify, SigEval) *being malleable with respect to a set of transformations* $\mathcal{T}$ *and having a simulator/extractor* (SimExtGen, SimSign, SigExt)*, is* simulation unforgeable*, if for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\epsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}(\mathsf{crs},\tau_s,\tau_e) \xleftarrow{R} \mathsf{SimExtGen}(1^\kappa),\\ (\mathsf{sk},\mathsf{pk}) \xleftarrow{R} \mathsf{KeyGen}(\mathsf{crs}),\\ (m^*,\sigma^*) \xleftarrow{R} \mathcal{A}^{\mathsf{SimSign}(\mathsf{crs},\tau_s,\mathsf{pk},\cdot)}(\mathsf{crs},\mathsf{pk},\tau_e)\\ (\vec{m},T) \leftarrow \mathsf{SigExt}(\mathsf{crs},\mathsf{pk},\tau_e,m^*,\sigma^*,\mathcal{Q})\end{array} : \begin{array}{l}\\ \\ (m^*,\sigma^*)\notin\mathcal{Q} \ \wedge\\ (\vec{m}\not\subseteq\mathcal{Q}_m \vee m^*\neq T(\vec{m}) \vee T\notin\mathcal{T})\end{array}\right] \leq \epsilon(\kappa),$$

where $\mathcal{Q}=\mathcal{Q}_m\times\mathcal{Q}_\sigma$ *is a list for keeping track of queried message-signature pairs.*

Definition 40 (Simulation Context-Hiding)**.** *A simulatable signature scheme* (Gen, KeyGen, Sign, Verify, SigEval) *being malleable with respect to a set of transformations* $\mathcal{T}$ *and having a simulator* (SimGen, SimSign)*, is* simulation context-hiding*, if for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\epsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}b\xleftarrow{R}\{0,1\},\ (\mathsf{crs},\tau_s)\xleftarrow{R}\mathsf{SimGen}(1^\kappa),\ (\mathsf{st},\mathsf{pk},\vec{m},\vec{\sigma},T)\xleftarrow{R}\mathcal{A}(\mathsf{crs},\tau_s),\\ \mathsf{Return}\ \bot\ \mathsf{if}\ (T\notin\mathcal{T}\ \vee\ \exists i:\mathsf{Verify}(\mathsf{crs},\mathsf{pk},m_i,\sigma_i)=0),\\ \mathsf{Else\ if}\ b=0:\sigma\xleftarrow{R}\mathsf{SimSign}(\mathsf{crs},\tau_s,\mathsf{pk},T(\vec{m})),\\ \mathsf{Else\ if}\ b=1:\sigma\xleftarrow{R}\mathsf{SigEval}(\mathsf{crs},\mathsf{pk},T,\vec{m},\vec{\sigma}),\\ b^*\xleftarrow{R}\mathcal{A}(\mathsf{state},\sigma)\end{array} : b=b^*\right] \leq \epsilon(\kappa).$$

2.3.2 Constructions

In [CKLM14], the authors give a simple, generic construction of malleable signatures for unary transformations from cm-NIZKs, which allows the to cover a large class of such transformations. The idea is to prove knowledge of the signing key and to include the message to-be-signed into the instance. More formally, the authors use a hard relation $\mathcal{R}_{\mathsf{pk}}$ generated by generator $\mathcal{G}$ and a cm-NIZK Π = (CRSSetup, Proof, Verify, ZKEval) that is malleable with respect to class $\mathcal{T}_{\mathsf{nizk}}$ and define a related relation $\mathcal{R}$ such that $(\mathsf{sk},\mathsf{pk})\in\mathcal{R}_{\mathsf{pk}} \Leftrightarrow (\mathsf{sk},(\mathsf{pk},m))\in\mathcal{R}$. Then, the construction of a simulatable signature scheme that is malleable with regard to a class $\mathcal{T}_{\mathsf{sig}}$ works as follows:

$\mathsf{Gen}(1^\kappa)$**:** Return $\mathsf{crs}\xleftarrow{R}\mathsf{CRSSetup}(1^\kappa)$.

$\mathsf{KeyGen}(\mathsf{crs})$**:** Compute $(\mathsf{pk}',\mathsf{sk}')\xleftarrow{R}\mathcal{G}(1^\kappa)$ and return $(\mathsf{pk},\mathsf{sk})\leftarrow(\mathsf{pk}',(\mathsf{pk}',\mathsf{sk}'))$.

$\mathsf{Sign}(\mathsf{crs},\mathsf{sk},m)$**:** Return $\sigma\xleftarrow{R}\Pi.\mathsf{Proof}(\mathsf{crs},(\mathsf{pk}',m),\mathsf{sk}')$.

$\mathsf{Verify}(\mathsf{crs},\mathsf{pk},\sigma,m)$**:** Return $\Pi.\mathsf{Verify}(\mathsf{crs},(\mathsf{pk},m),\sigma)$.

$\mathsf{SigEval}(\mathsf{crs},\mathsf{pk},T,m,\sigma)$**:** For $T\in T_{\mathsf{sig}}$, let $T_{\mathsf{inst}}(\mathsf{pk},m)=(\mathsf{pk},T(m))$ and $T_{\mathsf{wit}}=id$ and return $\sigma'\xleftarrow{R}\Pi.\mathsf{ZKEval}(\mathsf{crs},(T_{\mathsf{inst}},T_{\mathsf{wit}}),(\mathsf{pk},m),\sigma)$ as signature for $m'=T(m)$.

Not surprisingly, the properties of the cm-NIZK carry over to the above construction:

Theorem 5 ([CKLM14]). *If the cm-NIZK* Π *is ZK, then the above construction is a simulatable signature scheme.*

Theorem 6 ([CKLM14]). *If the cm-NIZK* Π *is strongly derivation private with respect to* $\mathcal{T}_{\mathsf{nizk}}$*, then the above construction is simulation context-hiding with respect to* $\mathcal{T}_{\mathsf{sig}}$*.*

Theorem 7 ([CKLM14]). *If the cm-NIZK* Π *is controlled-malleable simulation-sound extractable with respect to* $\mathcal{T}_{\mathsf{nizk}}$ *and* $\mathcal{R}_{\mathsf{pk}}$ *is a hard relation, then the above construction is simulation unforgeable with respect to* $\mathcal{T}_{\mathsf{sig}}$*.*

2.3.3 Applications

One main application—as outlined in [CKLM14]—is the construction of delegatable anonymous credentials (DACs). In contrast to conventional anonymous credential systems (ACs), DACs allow credential owners to delegate their credentials and do not require the credential issuer's public key during verification. Even more, DACs protect the identity of every link of the delegation chain and, thereby, make it infeasible to obtain any information about the credential owner. All in all, this gives stronger privacy guarantees, as the identity of the credential issuer—which could be some local authority—leaks already important information. As already mentioned above, malleable signatures for general transformation imply some other general concepts such as policy-based signatures [BF14] (cf. Section 2.2) as well as delegatable functional signatures [BMS13] (cf. Section 2.5) and thus also allow all the application of these schemes.

2.4 P-Homomorphic Signatures

The notion of P-homomorphic signatures has been introduced by Ahn et al. [ABC+12, ABC+15] as a means to formalize various types of signature schemes that allow to publicly compute on signed data. In a nutshell, for a predicate P their notion allows when given signatures on a message set M to publicly derive a signature on any message m' such that $P(M, m') = 1$. Subsequently to their introduction, [ALP12] strengthened the original privacy definition to also cover unlinkability issues. Deiseroth et al. [DFF+13] extended P-homomorphic signatures such that it is possible to combine schemes for different predicates into new schemes for logical formulas over these predicates (statically adjustable) or instead of fixing the predicate for a scheme to dynamically choose the predicate when singing a message (dynamically adjustable). We note, that the framework of P-homomorphic signatures has been generalized by malleable signatures for generic transformation covered in Section 2.3.

2.4.1 Formal Definitions

Before introducing the formal definition we need to fix some notation. Let $P : 2^{\mathcal{M}} \times \mathcal{M} \to \{0,1\}$ be a predicate and let us call a message m' *derivable* from $M \subseteq \mathcal{M}$ if $P(M, m') = 1$. Furthermore, let $P^i(M)$ be the set of messages derivable from $P^{i-1}(M)$ with $P^0(M) = \{m' \in \mathcal{M} \mid P(M, m') = 1\}$ and let us denote the set of messages derivable from M through iterated derivation as $P^*(M) := \bigcup_{i=0}^{\infty} P^i(M)$.

Definition 41 (P-Homomorphic Signatures [ABC+12, ALP12])**.** *A P-homomorphic signature scheme for a predicate $P : 2^{\mathcal{M}} \times \mathcal{M} \to \{0,1\}$ is a triple of algorithms* (PH.KeyGen, PH.SignDerive, PH.Verify) *which are defined as follows:*

$\mathsf{PH.KeyGen}(1^\kappa)$**:** *This algorithm takes a security parameter κ and outputs a key pair $(\mathsf{sk}, \mathsf{pk})$ (where the private key sk is treated as a signature on the empty message $\lambda \in \mathcal{M}$).*

$\mathsf{PH.SignDerive}(\mathsf{pk}, (\{\sigma_m\}_{m\in M}, M), m')$**:** *This (probabilistic) algorithm takes as input a public key pk, a set of messages $M \subseteq \mathcal{M}$, a corresponding set of signatures $\{\sigma_m\}_{m\in M}$ and a derived message $m' \in \mathcal{M}$. If $P(M, m') = 0$ it returns $\perp$ and otherwise a derived signature σ'.*

We use $\mathsf{PH.Sign}(\mathsf{sk}, m) := \mathsf{PH.SignDerive}(\mathsf{pk}, (\mathsf{sk}, \lambda), m, \cdot)$ to denote that any message can be derived when the original signature is the signing key.

$\mathsf{PH.Verify}(\mathsf{pk}, \sigma, m)$**:** *This deterministic algorithm takes as input a public key pk, a signature σ and a message m and outputs 1 if the signature is valid and 0 otherwise.*

For correctness one requires that for all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{PH.KeyGen}(1^\kappa)$, for any set $M \subseteq \mathcal{M}$, any message $m' \in \mathcal{M}$ such that $P(M, m') = 1$ we have that

$$\mathsf{PH.SignDerive}(\mathsf{pk}, (\mathsf{PH.Sign}(\mathsf{sk}, M), M), m') \neq \perp \text{ and}$$
$$\mathsf{PH.Verify}(\mathsf{pk}, m', \mathsf{PH.SignDerive}(\mathsf{pk}, \mathsf{PH.Sign}(\mathsf{sk}, M), M), m') = 1.$$

Before we define the security properties, we have to introduce some oracles and the lists $\mathcal{T}$ and $\mathcal{Q}$ to keep track of the oracle queries.

$\mathcal{O}^{\mathsf{Sign}}(\mathsf{sk}, m)$**:** Choose a handle h, run $\sigma \leftarrow \mathsf{PH.Sign}(\mathsf{sk}, m)$, add (h, m, σ) to $\mathcal{T}$ and return h.

$\mathcal{O}^{\mathsf{SignDerive}}(\vec{h}, m')$**:** Given $\vec{h} = (h_1, \ldots, h_k)$ and m', retrieve $\{(h_i, m_i, \sigma_i)\}_{i=1}^k$ from $\mathcal{T}$ and return $\perp$ if one of those do not exist. Otherwise, let $M \leftarrow (m_1, \ldots, m_k)$ and $\{\sigma_m\}_{m\in M} = (\sigma_1, \ldots, \sigma_k)$. If $P(M, m') \neq 1$ return $\perp$, else run $\sigma' \leftarrow \mathsf{PH.SignDerive}(\mathsf{pk}, (\{\sigma_m\}_{m\in M}, M), m')$, choose a handle h', store (h', m', σ') to $\mathcal{T}$ and return h'.

$\mathcal{O}^{\mathsf{Reveal}}(h)$**:** If no tuple of the form (h, m', σ') exists in $\mathcal{T}$ return $\perp$. Otherwise return σ' and add (m', σ') to $\mathcal{Q}$.

Definition 42 (Unforgeability)**.** *A P-homomorphic signature scheme is called unforgeable, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that*

$$\mathsf{Pr}\left[\begin{array}{c}((\mathsf{sk}, \mathsf{pk})) \leftarrow \mathsf{PH.Setup}(1^\kappa), \mathcal{O} \leftarrow \{\mathcal{O}^{\mathsf{Sign}}(\mathsf{sk}, \cdot), \mathcal{O}^{\mathsf{SignDerive}}(\cdot, \cdot), \mathcal{O}^{\mathsf{Reveal}}(\cdot)\}, \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}) : \mathsf{PH.Verify}(\mathsf{pk}, m^*, \sigma^*) = 1 \ \wedge \ m^* \notin P^*(M_{\mathcal{Q}})\end{array}\right] \leq \epsilon(\kappa) \ ,$$

where $M_{\mathcal{Q}}$ represents the set of messages $M \in \mathcal{M}$ that is contained within $\mathcal{Q}$.

Now, we are going to present privacy definitions for P-homomorphic signatures. Originally, [ABC+12] have defined strong context hiding as a replacement to the privacy notion denoted as weak context hiding in [BF11a] (which has been defined in context of homomorphic signatures

for polynomial functions and is adapted from [BBD+10]). Later, [ALP12] introduced a stronger notion denoted as completely context hiding. While these notions require statistical closeness of distributions, there are also game based security notions, which can be seen as the computational equivalents to the ones mentioned before. We do not revisit the computational ones and refer the readers to [ABC+12, ALP12] for these definitions.

Definition 43 (Strong Context Hiding [ABC+12])**.** *A P-homomorphic signature scheme is strongly context hiding if for all keys* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{PH.KeyGen}(1^\kappa)$*, for all messages* $M \subseteq M^*$ *and* $m' \in \mathcal{M}$ *such that* $P(M, m') = 1$*, the following two distributions are statistically close*

$$\{(\mathsf{sk}, \{\sigma_m\}_{m\in M} \leftarrow \mathsf{PH.Sign}(\mathsf{sk}, M), \mathsf{PH.Sign}(\mathsf{sk}, m'))\}_{\mathsf{sk},M,m'},$$
$$\{(\mathsf{sk}, \{\sigma_m\}_{m\in M} \leftarrow \mathsf{PH.Sign}(\mathsf{sk}, m'), \mathsf{PH.SignDerive}(\mathsf{pk}, (\{\sigma_m\}_{m\in M}, M), m'))\}_{\mathsf{sk},M,m'}.$$

Strong context hiding states that a derived signature on m' (from an honestly-generated original signature) is statistically indistinguishable from a fresh signature on m'. This implies that a derived signature on m' is indistinguishable from a signature generated independently of M. We note that in contrast to strong context hiding, weak context hiding [BF11a] only ensures privacy when the original signatures remain hidden.

While, however, strong context hiding only considers honestly generated signatures, [ALP12] explicitly considers also maliciously generated ones (i.e., signatures that pass the verification algorithm, but may not have been computed honestly) with a notion called complete context hiding.

Definition 44 (Complete Context Hiding [ALP12])**.** *A P-homomorphic signature scheme is completely context hiding for predicate P if for all keys* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{PH.KeyGen}(1^\kappa)$*, for all messages* $M \subseteq M^*$ *and* $m' \in \mathcal{M}$ *such that* $P(M, m') = 1$*, for all* $\{\sigma_m\}_{m\in M}$ *such that* $\mathsf{PH.Verify}(\mathsf{pk}, M, \{\sigma_m\}_{m\in M}) = 1$ *the following two distributions are statistically close*

$$\{(\mathsf{sk}, \mathsf{PH.Sign}(\mathsf{sk}, M))\}_{\mathsf{sk},M,m'}, \quad \{(\mathsf{sk}, \mathsf{PH.SignDerive}(\mathsf{pk}, (\{\sigma_m\}_{m\in M}, M), m'))\}_{\mathsf{sk},M,m'}.$$

We note that one can even strengthen this definition further by letting the adversary generate the keys (as e.g., done in a somewhat related approach [FHS15] or in Section 2.3).

Adjustable P-Homomorphic Signature Schemes. In [DFF+13], Deiseroth et al. studied to which extent it is possible to adjust predicates in P-homomorphic signatures. Recall, that the predicate P is fixed for a given scheme. Adjustable here means firstly to *statically* combine schemes for fixed predicates into a scheme for more expressive ones. In particular, schemes for logical combinations (AND, OR, NOT) of single predicates. Secondly, to *dynamically* decide which predicate to use when signing a message.

For the *static* case, [DFF+13] show that the "naive" canonical approach for AND works, i.e., given schemes for predicates $P_1, \ldots, P_n$ to construct a scheme for predicate $\bigwedge_{i=1}^{n} P_i$ that works component wise. This means that one signs each message with the schemes for predicates $P_1, \ldots, P_n$ individually and derives signatures by applying the corresponding algorithms for

each components. Unfortunately, [DFF+13] prove that such a canonical construction cannot exist for OR and NOT.

For the *dynamic* case, [DFF+13] present a generic "certify-then-sign" construction. Basically, they use a conventional signature scheme to certify a public key of a P-homomorphic signature scheme for predicate P and sign the message under the secret key of the P-homomorphic scheme. In order to achieve context hiding, however, it is important that every time a message is signed with respect to a specific predicate P, the key pair for the P-homomorphic scheme needs to be identical, which can for instance be achieved by pseudorandomly generating the key pairs.

2.4.2 Constructions

We are going to see various different signature schemes in Section 4 and Section 5 that can be modeled or investigated within the framework of P-homomorphic signatures. Consequently, we will not explicitly present any instantiation here and refer the reader to Section 4 and Section 5.

For the sake of completeness, however, we mention that they can be used to compute on authenticated data and prominent classes of schemes are linearly homomorphic signatures [BFKW09, BF11b, Fre12, CFW12, ALP13, LPJY13, WHW13, CMP14], homomorphic signatures for polynomial functions [BF11a, HMO13, CFW14], fully homomorphic signatures [BFS14, GVW14] as well as quotable/redactable signatures [JMSW02, ABC+12, ABC+15]. Furthermore, the concept of structure-preserving signatures on equivalence classes [HS14, FHS14] is also related to this framework.

2.4.3 Applications

Like with the constructions, we defer the reader to Section 4 and Section 5 for a comprehensive discussion.

2.5 Operational Signatures

Operational signature (OS) schemes have been introduced by Backes et al. [BDF+14] as an even more general concept then all the aforementioned general frameworks[9]. Namely, [BDF+14] argue that functional signatures (cf. Section 2.1) as well as policy-based signatures (cf. Section 2.2) do not consider that the input (message) may be a set of already signed messages. Moreover, they argue that P-homomorphic signatures (cf. Section 2.4) and malleable signatures for general transformations (cf. Section 2.3) do not consider any requirements on the keys in their predicates and transformations are all under the same key respectively. Consequently, [BDF+14] aims to bridge the gap between functional keys and malleable signatures.

[9] Operational signatures (OS) are a further generalization of so called delegatable functional signatures (DFS) [BMS13] introduced by some of the authors of [BDF+14] before. Due to OS being more general we omit an explicit description of DFS here.

2.5.1 Formal Definitions

Before we introduce OS schemes, we require some notational preliminaries. OS are defined for ensembles $\mathcal{P}$ of predicates P (which basically describe the admissible operations). These n-ary predicates P operate on n pairs $(id, m) \in \mathcal{ID} \times \mathcal{M}$ of key identifiers and messages. A key identifier id can thereby be viewed as a public handle to a secret sk_{id} or public key pk_{id}. More precisely, from the n pairs, one considers the first $n-2$ pairs $(id_{\mathsf{in},i}, m_{\mathsf{in},i})_{i=1}^{n-2}$ as the input pairs and the last two pairs $(id_{\mathsf{eval}}, m_{\mathsf{eval}})$ and $(id_{\mathsf{out}}, m_{\mathsf{out}})$ as the evaluation and the output pair respectively. This entire sequence is abbreviated as $(\mathbf{id}, \mathbf{m})$. We note that OS schemes are defined in a way that they cover message authentication codes (MACs) as well as signature schemes. However, we only consider signatures, i.e., the public key setting, and will not consider MACs, i.e., the secret key setting, subsequently.

Now, informally, such a predicate P upon input $(\mathbf{id}, \mathbf{m})$ should return 1 if one holds signatures (authenticators) for each $m_{\mathsf{in},i}$ which verify under the cryptographic keys under handles $id_{\mathsf{in},i}$ and if one evaluates these messages under key id_{eval} to m_{eval}, then one derives a signature (authenticator) for m_{out}, which verifies under the key to handle id_{out}. We stress that the predicate P only works on key-message pairs (m, id) and not on signatures. The actual transformation of the signatures will be carried out by the $\mathsf{OS.Eval}$ algorithm which in addition takes signatures $\sigma_{\mathsf{in},i}$ as well as a pair $(\mathsf{k}_{\mathsf{eval}}, m_{\mathsf{eval}})$ instead of $(id_{\mathsf{eval}}, m_{\mathsf{eval}})$, i.e., the handle to the eval key is replaced by the actual key $\mathsf{k}_{\mathsf{eval}}$. The sequence $((id_{\mathsf{in},i}, m_{\mathsf{in},i}, \sigma_{\mathsf{in},i})_{i=1}^{n-2}, (\mathsf{k}_{\mathsf{eval}}, m_{\mathsf{eval}}), (id_{\mathsf{out}}, m_{\mathsf{out}}))$ which is input to the $\mathsf{OS.Eval}$ algorithm is abbreviated as $(\mathbf{id}[\mathsf{k}_{\mathsf{eval}} \to \mathbf{id}_{\mathsf{eval}}], \mathbf{m}, \sigma)$.

Definition 45 (Operational Signatures [BDF+14])**.** *An operational signature (OS) scheme for predicates $\mathcal{P}$ consists of four PPT algorithms* $(\mathsf{OS.Setup}, \mathsf{OS.KeyGen}, \mathsf{OS.Eval}, \mathsf{OS.Verify})$ *which are defined as follows:*

- $\mathsf{OS.Setup}(1^\kappa)$: *This setup algorithm takes a security parameter κ and outputs a master secret key* msk *and some public parameters* pp *(where it is assumed that* pp *is an implicit input to all other algorithms).*
- $\mathsf{OS.KeyGen}(\mathsf{msk}, id)$: *This probabilistic key generation algorithm takes as input a master secret key* msk *and some $id \in \mathcal{ID}$ and outputs a key* k_{id}*.*
- $\mathsf{OS.Eval}((\mathbf{id}[\mathsf{k}_{\mathsf{eval}} \to \mathbf{id}_{\mathsf{eval}}], \mathbf{m}, \sigma), P)$: *This (probabilistic) signature algorithm takes as input a sequence $(\mathbf{id}[\mathsf{k}_{\mathsf{eval}} \to \mathbf{id}_{\mathsf{eval}}], \mathbf{m}, \sigma)$ consisting of triples $(id_{\mathsf{in},i}, m_{\mathsf{in},i}, \sigma_{\mathsf{in},i})_{i=1}^{n-2}$, the cryptographic key $\mathsf{k}_{\mathsf{eval}}$ for evaluating message m_{eval}, the pair of target values $(id_{\mathsf{out}}, m_{\mathsf{out}})$, and (the description of) a predicate P. The algorithm outputs a signature σ in a set $\mathcal{S}$ or a special symbol $\perp \notin \mathcal{S}$.*
- $\mathsf{OS.Verify}(\mathsf{k}_{\mathsf{out}}, m_{\mathsf{out}}, \sigma)$: *This (deterministic) verification algorithm takes an operational key $\mathsf{k}_{\mathsf{out}}$, a message $m_{\mathsf{out}} \in \mathcal{M}$ and a signature σ and outputs 1 if the signature is valid or 0 otherwise.*

For correctness of an OS for predicates $\mathcal{P}$ one requires that for any $P \in \mathcal{P}_\kappa$, all message tuples $\mathbf{m} = (m_{\mathsf{in},1}, \ldots, m_{\mathsf{in},n-2}, m_{\mathsf{eval}}, m_{\mathsf{out}}) \in \mathcal{M}_\kappa^n$, all key handles $\mathbf{id} = (id_{\mathsf{in},1}, \ldots, id_{\mathsf{in},n-2}, id_{\mathsf{eval}}, id_{\mathsf{out}}) \in$

$\mathcal{ID}_\kappa^n$ and all signatures $\sigma = (\sigma_1, \ldots, \sigma_{n-2}) \in \mathcal{S}_\kappa^{n-2}$ where $n \geq 2$ is the arity of P, one has

$$\Pr\left[\begin{array}{l|l} \mathsf{OS.Verify}(\mathsf{k_{out}}, m_{\mathsf{out}}, \sigma_{\mathsf{out}}) = & (\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{OS.Setup}(1^\kappa), \\ P(\mathbf{m}, \mathbf{id}) & (\mathsf{k_{id}})_{id \in \mathbf{id}} \leftarrow \mathsf{OS.KeyGen}(\mathsf{msk}, id)_{id \in \mathbf{id}}, \\ & \sigma_{\mathsf{out}} \leftarrow \mathsf{OS.Eval}((\mathbf{id}[\mathsf{k_{eval}} \rightarrow \mathbf{id}_{\mathsf{eval}}], \mathbf{m}, \sigma), P) \\ & \forall i \in [n-2] : \mathsf{OS.Verify}(\mathsf{k}_{\mathsf{in},i}, m_{\mathsf{in},i}, \sigma_{\mathsf{in},i}) = 1 \end{array}\right] \leq 1 - \epsilon(\kappa) \ ,$$

where $\epsilon(\cdot)$ is a negligible function and the probability is taken over the coin tosses of $\mathsf{OS.Setup}$, $\mathsf{OS.KeyGen}$ and $\mathsf{OS.Eval}$.

Furthermore, as within conventional signature schemes one requires unforgeability under adaptive chosen message attacks and the additional *privacy* property (the latter is basically an adapted notion of context hiding from [ALP12, ALP13, DFF+13] in context of P-homomorphic signatures (cf. Section 2.4)).

In order to state these security notions, we need to define some oracles and introduce two lists $\mathcal{QID}$ and $\mathcal{QE}$. We note again that as we are only considering signatures but no MACs. Thus, in contrast to the notion in [BDF+14], we do neither require a Verify' oracle (as verification is always public) nor the notions of weak- as well as outsider-privacy (but only privacy).

$\mathcal{O}^{\mathsf{KeyGen}'}(\mathsf{msk}, id)$ takes a master secret key and an identifier id. It adds id to $\mathcal{QID}$ and returns $\mathsf{OS.KeyGen}(\mathsf{msk}, id)$.

$\mathcal{O}^{\mathsf{Eval}'}(\mathsf{pp}, (\mathbf{id}, \mathbf{m}, \sigma), P)$ adds $(\mathbf{id}, \mathbf{m}, \sigma)$ to $\mathcal{QE}$ and returns $\mathsf{OS.Eval}(\mathsf{pp}, (\mathbf{id}[\mathsf{k_{eval}} \rightarrow \mathbf{id}_{\mathsf{eval}}], \mathbf{m}, \sigma), P)$.

$\mathcal{O}^{\mathsf{Ch}_b}((\mathbf{id^0}, \mathbf{m^0}, \sigma^0), (\mathbf{id^1}, \mathbf{m^1}, \sigma^1), P_0, P_1)$ works as follows:

- if $P_0(\mathbf{id^0}, \mathbf{n^0}) = 0$ or $P_1(\mathbf{id^1}, \mathbf{n^1}) = 0$ return $\perp$;
- if $(\mathsf{k}^0_{\mathsf{out}}, m^0_{\mathsf{out}}) \neq (\mathsf{k}^1_{\mathsf{out}}, m^1_{\mathsf{out}})$ return $\perp$;

compute $\sigma^0_{\mathsf{eval}} \leftarrow \mathsf{Eval}'(\mathsf{pp}, (\mathbf{id^0}, \mathbf{m^0}, \sigma^0), P_0)$ and $\sigma^1_{\mathsf{eval}} \leftarrow \mathsf{Eval}'(\mathsf{pp}, (\mathbf{id^1}, \mathbf{m^1}, \sigma^1), P_1)$. If $\sigma^0_{\mathsf{eval}} = \perp$ or $\sigma^1_{\mathsf{eval}} = \perp$ return $\perp$. Else return σ^b_{eval}.

Before presenting the security games, we need two additional concepts. Firstly, we need to discuss when a predicate is called *well-formed*. Therefore, we consider the basic signature predicate P_{sig} where we let $\mathcal{ID} = \{id_{\mathsf{sig}}, id_{\mathsf{pub}}\}$ with $id_{\mathsf{sig}} \in \mathcal{ID} \setminus \mathcal{ID}_{\mathsf{pub}}$ being the secret signing key and $id_{\mathsf{pub}} \in \mathcal{ID}_{\mathsf{pub}}$ being the public verification key. The predicate P_{sig} for a simple signature functionality is

$$P_{sig}(\mathbf{id}, \mathbf{m}) = \begin{cases} 1 \text{ if } id_{\mathsf{eval}} = id_{\mathsf{sig}} \in \mathcal{ID}, id_{\mathsf{out}} = id_{\mathsf{pub}} \in \mathcal{ID}, m_{\mathsf{eval}} = m_{\mathsf{out}} \in \mathcal{M} \\ 0 \text{ otherwise.} \end{cases}$$

Now, a predicate ensemble $\mathcal{P}$ is called *well-formed* if $P_{sig} \in \mathcal{P}$.

Moreover, we need to define what trivial identifier-message pairs are. Basically, the set $\mathsf{Triv}(\mathcal{QID}, \mathcal{QE}) \subseteq \mathcal{ID} \times \mathcal{M}$ is the following set:

Case 1 (Base Case): for any $(\mathbf{id}, \mathbf{m}, P) \in \mathcal{QE}$ we have $(id_{\mathsf{out}}, m_{\mathsf{out}}) \in \mathsf{Triv}(\mathcal{QID}, \mathcal{QE})$.

Case 2 (Recursion): if $(id_{\mathsf{in},i}, m_{\mathsf{in},i}) \in \mathsf{Triv}(\mathcal{QID}, \mathcal{QE})$ for each i, and $id_{\mathsf{eval}} \in \mathcal{QID}$, then also any $(id_{\mathsf{out}}, m_{\mathsf{out}})$ for which there is some $P \in \mathcal{P}$ and some m_{eval} such that $P(\mathbf{id}, \mathbf{m}) = 1$, is in $\mathsf{Triv}(\mathcal{QID}, \mathcal{QE})$.

Definition 46 (Unforgeability). *An OS scheme for predicates $\mathcal{P}$ over $\mathcal{ID}_{pub}$, $\mathcal{ID}$, $\mathcal{M}$ is unforgeable (EUF-CMA secure), if for any (efficient) algorithm $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that*

$$\mathsf{Pr}\left[\begin{array}{c} \mathcal{QE} \leftarrow \emptyset, \mathcal{QID} \leftarrow \mathcal{ID}_{pub}, \mathcal{O} \leftarrow \{\mathcal{O}^{\mathsf{KeyGen}'}(\mathsf{msk}, \cdot), \mathcal{O}^{\mathsf{Eval}'}(\cdot,\cdot,\cdot)\}, \\ (id^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}) : (\mathsf{pp}, id^*, m^*, \sigma) \in \mathcal{QE} \ \wedge \\ \mathsf{Verify}(\mathsf{k}_{\mathsf{id}^*}, \mathsf{m}^*, \sigma^*) = 1 \ \wedge \ (id^*, m^*) \notin \mathsf{Triv}(\mathcal{QID}, \mathcal{QE}) \end{array}\right] \leq \epsilon(\kappa) \ ,$$

where the last condition, i.e., $(id^*, m^*) \notin \mathsf{Triv}(\mathcal{QID}, \mathcal{QE})$) needs to hold at the point in time when the Verify *query has been made by $\mathcal{A}$.*

Definition 47 (Privacy). *An OS scheme for well-formed predicates $\mathcal{P}$ over $\mathcal{ID}_{pub} \subseteq \mathcal{ID}$, $\mathcal{M}$ is private, if for any (efficient) algorithm $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that*

$$\mathsf{Pr}\left[\begin{array}{c} \mathcal{QID} \leftarrow \emptyset, b \xleftarrow{R} \{0,1\}, (\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{OS.Setup}(1^\kappa), \\ \mathcal{O} \leftarrow \{\mathcal{O}^{\mathsf{KeyGen}}(\mathsf{msk}, \cdot), \mathcal{O}^{\mathsf{Ch}_b}(\cdot,\cdot,\cdot,\cdot)\}, b^* \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}) : b^* = b \end{array}\right] - \frac{1}{2} \leq \epsilon(\kappa) \ ,$$

2.5.2 Constructions

Let $(\mathsf{DKeyGen}, \mathsf{DSign}, \mathsf{DVerify})$ be a digital signature scheme, $i\mathcal{O}$ be an indistinguishability obfuscator and H a hash function.

$\mathsf{OS.Setup}(1^\kappa)$**:** Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{DKeyGen}(1^\kappa)$, generate an obfuscation $c \leftarrow i\mathcal{O}(C_{H,\mathsf{sk},\mathsf{pk}})$ of the circuit $C_{H,\mathsf{sk},\mathsf{pk}}$ (cf. Figure 2). Set $\mathsf{pp} \leftarrow (c, \mathsf{pk})$ and the master key as $\mathsf{msk} \leftarrow (\mathsf{sk}, \mathsf{pp})$.

$\mathsf{OS.KeyGen}(\mathsf{msk}, id)$**:** On input a master secret $\mathsf{msk} = (\mathsf{sk}, \mathsf{pp})$ and an identifier $id \in \mathcal{ID}$, this algorithm returns the pair (id, σ_{id}), where $\sigma_{id} \leftarrow \mathsf{DSign}(id, \mathsf{sk})$.

$\mathsf{OS.Eval}((\mathbf{id}[\mathsf{k}_{\mathsf{eval}} \rightarrow \mathbf{id}_{\mathsf{eval}}], \mathbf{m}, \sigma), P)$**:** On input $\mathsf{pp} = (c, \mathsf{pk})$, a tuple $(\mathbf{id}[\mathsf{k}_{\mathsf{eval}} \rightarrow \mathbf{id}_{\mathsf{eval}}], \mathbf{m}, \sigma)$ consisting of triples $((id_{\mathsf{in},i}, m_{\mathsf{in},i}, \sigma_{\mathsf{in},i})_{i=1}^{n-2}$, the cryptographic key $\mathsf{k}_{\mathsf{eval}} = (\mathsf{pk}, id_{\mathsf{eval}}, \sigma_{\mathsf{eval}})$ for evaluating message m_{eval}, the pair of target values $(id_{\mathsf{out}}, m_{\mathsf{out}})$, and a predicate P, compute first the digest $h \leftarrow H((\mathbf{id}[\mathsf{k}_{\mathsf{eval}} \rightarrow \mathbf{id}_{\mathsf{eval}}], \mathbf{m}, \sigma), P)$ and then the signature $\sigma \leftarrow c(((\mathbf{id}[\mathsf{k}_{\mathsf{eval}} \rightarrow \mathbf{id}_{\mathsf{eval}}], \mathbf{m}, \sigma), P), h)$.

$\mathsf{OS.Verify}(\mathsf{k}_{id}, m, \sigma)$**:** On input a key $\mathsf{k}_{id} = (\mathsf{pp}, id, \sigma_{id})$, where $\mathsf{pp} = (c, \mathsf{pk})$ a message m and a signature σ, return $\mathsf{DVerify}(\mathsf{pk}, m \| id, \sigma)$.

Theorem 8 ([BDF+14]). *Let $(\mathsf{DKeyGen}, \mathsf{DSign}, \mathsf{DVerify})$ be an unforgeable and deterministic signature scheme, $i\mathcal{O}$ be a random-oracle based indistinguishability obfuscator for a class of upstream hashing-only circuits $\mathcal{C}$ containing $C_{H,\mathsf{sk},\mathsf{pk},C}$ and C_{fake} (cf. Figure 2 in [BDF+14]),* PRF *is a pseudorandom function and H be a hash function modeled as a random oracle. Then the above construction is an unforgeable and private* OS *scheme for all predicates $\mathcal{P}$ of fixed polynomial size such that every $P \in \mathcal{P}$ is efficiently computable.*

$C_{H,\mathsf{sk},\mathsf{pk},\mathcal{P}}(((\mathbf{id}[\mathsf{k}_{\mathsf{eval}} \to \mathbf{id}_{\mathsf{eval}}], \mathbf{m}, \sigma), P), h)$:
If $h = H((\mathbf{id}[\mathsf{k}_{\mathsf{eval}} \to \mathbf{id}_{\mathsf{eval}}], \mathbf{m}, \sigma), P)$
$\wedge\ \forall i : \mathsf{DVerify}(\mathsf{pk}, m_{\mathsf{in},i} \| id_{\mathsf{in},i}, \sigma_i) = 1$
$\wedge\ \mathsf{DVerify}(\mathsf{pk}, \mathsf{k}_{\mathsf{eval}}, \sigma_{\mathsf{eval}}) = 1$
$\wedge\ P(\mathbf{id}, \mathbf{m}) = 1$
output $\sigma_{\mathsf{out}} \leftarrow \mathsf{DSign}(\mathsf{sk}, m_{\mathsf{out}} \| id_{\mathsf{out}})$
else
output $\perp$

Figure 2: Obfuscated circuit $C_{H,\mathsf{sk},\mathsf{pk},\mathcal{P}}$

2.5.3 Applications

Since operational signatures are the most generic framework that covers numerous other frameworks such as P-homomorphic signatures, functional signatures as well as policy-based signatures and different types of signatures such as sanitizable signatures, redactable signatures, ring signatures, fully homomorphic signatures, we do not explicitly discuss any applications here.

3 Functional Signatures

In this section we provide an overview of signature schemes that allow to delegate signature generation to other parties for a class of messages meeting certain conditions. We emphasize that we call all of them functional signatures and this must not be confused with the *framework of functional digital signatures* discussed in Section 2.1. Unfortunaltely, the framework has a very generic name but not all signatures that fall under functional signatures can be covered by this framework (for instance, attribute-based signatures do not straightforwardly fit to this framework and proxy signatures have interactive delegation). Consequently, when we speak of functional signatures we mean what we consider as this class of signatures and not the framework.

3.1 Proxy Signatures

Proxy signatures (PS), firstly introduced in [MUO96], allow a party A to delegate its signing rights to some party B such that B can issue signatures on behalf of A. In this context, virtually all existing schemes implement the so called delegation by certificate approach. Here, A issues a certificate containing the description of the "allowed" message space $\mathcal{M}$ for B and B's public key. B then issues signatures with respect to this certificate.[10] Verification is performed by verifying the signatures of B, the certificate and whether the signed message is in line with $\mathcal{M}$.

3.1.1 Formal Definitions

While none of the early proxy signature schemes provides a formal security model, a security model was introduced in [BPW12]. This model is considered to be the standard model for PS and is based upon standard digital signature schemes. However, some instantiations [BPW12, MOY04] also make use of aggregate signature schemes [LMRS04] to reduce the size of the signature. Subsequently, we recall the model of [BPW12].[11]

$(\mathsf{D}(\mathcal{M}, \mathsf{pk}_i, \mathsf{sk}_i, j, \mathsf{pk}_j), \mathsf{P}(\mathsf{pk}_j, \mathsf{sk}_j, \mathsf{pk}_i))$**:** The originator and the proxy jointly compute a delegation for the message space $\mathcal{M}$ as well as a proxy signing key skp. The originator runs D and outputs the delegation σ computed using its signing key sk_i, whereas the proxy verifies the delegation and obtains the proxy signing key skp, which consists of its private signing key sk_j and the originators delegation.

$\mathsf{PS}(\mathsf{skp}, M)$**:** This algorithm computes and outputs a proxy signature σ_P for message $M \in \mathcal{M}$ using the proxy signing key skp.

$\mathsf{PV}(\mathsf{pk}_i, M, \sigma_P)$**:** This algorithm verifies whether proxy signature σ_P is a valid proxy signature for message M under pk_j, delegated by pk_i. On success, this algorithm outputs 1, and 0 otherwise.

[10] This description of the message space can be seen as the delegated function f.
[11] Note that we use a more compact representation of the model, which we adapt from [DHS14].

$\mathsf{ID}(\sigma_P)$**:** This algorithm outputs the identity j of the proxy, when given a proxy signature σ_P.

Definition 48 (Unforgeability)**.** *For all PPT adversaries $\mathcal{A}$ there exists a negligible function ϵ such that*

$$\Pr\left[\begin{array}{c}(\mathsf{sk}_1,\mathsf{pk}_1)\leftarrow\mathsf{DKeyGen}(\kappa),((M,\sigma),(M,\sigma_p,\mathsf{pk}_i))\leftarrow\mathcal{A}^{\mathcal{O}_R,\mathcal{O}_D,\mathcal{O}_E,\mathcal{O}_S,\mathcal{O}_{PS}}(\mathsf{pk}_1):\\ (\mathsf{DVerify}(\sigma,M,\mathsf{pk}_1)=1\wedge M\notin\mathcal{Q}_S)\ \vee\\ (\mathsf{PV}(\mathsf{pk}_i,M,\sigma_P)=1\ \wedge\ i\neq 1\ \wedge\ \mathsf{ID}(\sigma_P)=1\ \wedge\ M\notin\mathcal{Q}_{PS})\ \vee\\ (\mathsf{PV}(\mathsf{pk}_i,M,\sigma_P)=1\ \wedge\ i=1\ \wedge\ \mathsf{ID}(\sigma_P)=1\ \wedge\ M\notin\mathcal{Q}_{PS})\ \vee\\ (\mathsf{PV}(\mathsf{pk}_i,M,\sigma_P)=1\ \wedge\ i=1\ \wedge\ \forall\mathcal{M}_{\mathsf{ID}(\sigma_P)}:M\notin\mathcal{M}_{\mathsf{ID}(\sigma_P)})\end{array}\right]\leq\epsilon(\kappa)$$

where the adversary has access to an oracle $\mathcal{O}_R$ to register public keys for users, to $\mathcal{O}_D$ for the designation of signing rights for message spaces $\mathcal{M}$, to $\mathcal{O}_E$ which exposes the ℓ-th proxy signing key produced during self-delegation (user 1 to user 1), to $\mathcal{O}_S$ to obtain standard signatures from user 1 and to a proxy sign oracle $\mathcal{O}_{PS}$ for proxy signatures by user 1. Furthermore, $\mathcal{Q}_S$ and $\mathcal{Q}_{PS}$ are the lists of queried standard and proxy signatures, respectively. With σ we denote a standard signature.

3.1.2 Instantiations

Since the introduction of PS a wide variety of PS schemes was proposed. We subsequently discuss the most relevant schemes in our context [SMP08, HS13b, BPW12, FP08]. Those schemes mostly differ in the way the description of the designated message space is encoded, and, thus, the size of the signatures and the computational costs for verification. State-of-the-art encodings include enumerating all allowed messages [SMP08], vector commitments with selective openings [HS13b], descriptions of deterministic PT Turing machines [BPW12], or simply abstract descriptions of the designated message space. In addition, [HS13b] also mentioned context-free grammars and regular expressions as possible encodings.

Besides the differences mentioned above, [BPW12, SMP08] employ aggregate signatures to reduce the size of proxy signatures. Table 1 compares the schemes mentioned above regarding their computational efficiency and signature size.

Scheme	Max. n	|D|	|PS|	|PV|	$\lvert\sigma\rvert$
DSS [BPW12]	1	$\mathcal{O}(\lvert TM\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(T(TM(\cdot)))$	$\mathcal{O}(\lvert TM\rvert)$
AGG [BPW12]	1	$\mathcal{O}(\lvert TM\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(T(TM(\cdot)))$	$\mathcal{O}(\lvert TM\rvert)$
[SMP08]	-	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(\max\{n,\lvert\mathcal{M}\rvert\})$	$\mathcal{O}(\max\{n,\lvert\mathcal{M}\rvert\})$
PC [HS13b]	1	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
MT [HS13b]	1	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\log\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\log\lvert\mathcal{M}\rvert)$

Table 1: Comparison of PS schemes. Here, n denotes the maximum number of consecutive delegations, $|\sigma|$ denotes the signature size, |D|, |PS|, |PV| denote the computational complexity of the respective algorithm, DSS denotes a digital signature scheme, AGG denotes an aggregate signature scheme, PC denotes a polynomial commitment scheme, MT denotes a Merkle-tree-based vector commitment, $|TM|$ denotes the size of the description of the deterministic PT Turing machine, $T(TM(\cdot))$ denotes the maximum number of steps TM takes to verify whether an arbitrary message m is an "allowed" message and $|\mathcal{M}|$ denotes the size of the message space.

3.1.3 Extensions

Subsequently, we discuss some extensions to the above presented model of proxy signatures.

Privacy in the Context of PS. Besides the variations in the expressiveness of the used encoding, an appropriate encoding may also deliver useful additional security properties [HS13b], i.e., privacy. This notion prevents the verifier from learning the whole designated messages space upon verification of a proxy signature on a single message. Subsequently, we recall this notion (adopted from [DHS14]):

Definition 49 (Privacy). *For all PPT adversaries $\mathcal{A}$ there exists a negligible function ϵ such that*

$$\mathsf{Pr}\left[\begin{array}{c}(\mathsf{sk}_1,\mathsf{pk}_1)\leftarrow\mathsf{DKeyGen}(\kappa),((i,c),\mathsf{state})\leftarrow\mathcal{A}^{\mathcal{O}_R,\mathcal{O}_D,\mathcal{O}_E,\mathcal{O}_S,\mathcal{O}_{PS}}(\mathsf{pk}_1),\\ \mathsf{skp}^*=(\mathsf{D}(\mathcal{M}_r,\mathsf{pk}_1,\mathsf{sk}_1,i,\mathsf{pk}_i),\mathsf{P}(\mathsf{pk}_i,\mathsf{sk}_i,\mathsf{pk}_1))\\ \mathcal{M}^*\leftarrow\mathcal{A}^{\mathcal{O}_R,\mathcal{O}_D,\mathcal{O}_E,\mathcal{O}_S,\mathcal{O}_{PS},\mathcal{O}_{PS'}}(\mathsf{state}):\mathcal{M}^*=\mathcal{M}_r\end{array}\right]\leq\frac{1}{|\mathbb{M}'|}+\epsilon(\kappa)$$

where the adversary has access to the same oracles as in the unforgeability game in the first phase. The adversary then outputs $c>1$ and i, a random warrant $\mathcal{M}_r$ of size $c+1$ is chosen from a message space $\mathbb{M}$ and a delegation of user i computed. Then, in the second phase the adversary has access to the same oracles as in phase 1 and an additional oracle $\mathcal{O}_{PS'}$ to obtain proxy signatures w.r.t. skp^* for randomly chosen messages from $\mathcal{M}_r$ at most c times. Note, that the adversary has no direct access to skp^* and thus is not aware of $\mathcal{M}_r$. The goal of the adversary is to guess the warrant $\mathcal{M}_r$ with negligible probability away from $\frac{1}{|\mathbb{M}'|}$ where $\mathbb{M}'$ represents the space of all potential messages $\mathbb{M}$ minus all message queried to $\mathcal{O}_{PS'}$.

This property can be reached by using a hiding vector commitment with selective openings to commit to the messages in the message space [HS13b]. In [HS13b], two instantiations based on polynomial commitments [KZG10] and randomized Merkle hash trees, respectively, are provided. Using the former approach, one obtains constant size signatures as well as constant verification costs, whereas one obtains signature sizes and verification costs logarithmic in the size of the message space when using the latter approach.

Hierarchical Proxy Signatures. [MOY04] extends the model of PS to hierarchical PS, that allow to model chains of delegation. Based on this model, they show the equality of hierarchical PS to key-insulated signatures [DKXY03]. In their paper, they also unveiled an issue regarding self-delegation in the original model. However, this issue was fixed in the extended version of [BPW12] and the fix is also included in the model presented above.

A Stronger Security Model. In [SMP08], the model of PS was extended to provide stronger security guarantees (security in their model implies security in the model of [BPW12]). In particular, they remove the requirement that proxy signing keys need to be registered via $\mathcal{O}_R$. Additionally, they emphasize the importance of an explicit description of the semantic of the warrant, as otherwise this could be exploited to obtain a certain degree of malleability. Furthermore, they also consider hierarchical PS and provide a hierarchical PS instantiation

in their strengthened model. This instantiation is based on sequential aggregate signatures [LMRS04]. Essentially, their approach is to "concatenate" the signed messages as well as the designation certificates by means of a sequential aggregate signature. The signature size in their construction is linear in the maximum of the number of delegations and the number of intermediate delegatees, respectively.

Anonymous Proxy Signatures. Anonymous proxy signatures [FP08] aim at hiding the identity of all intermediate delegatees in a proxy signature. To this end, the identification algorithm can no longer be publicly executed, but there is a dedicated opening authority (similar to the opening authority in group signatures) that can open signatures, and, therefore, identify delegatees. This additionally imposes the requirements of *traceability*, i.e., every valid signature can be correctly opened, and *non-frameability*, i.e., no user can be falsely accused of being a delegator/signer. Their scheme is based on a digital signature scheme, a public-key encryption scheme and a simulation-sound NIZK proof system for an NP-language. A proxy signature consists of an encryption of a conventional proxy signature together with a non-interactive proof for the validity of the encrypted statement.

3.2 Blank Digital Signatures

Blank Digital Signatures (BDS) [HS13a] are somewhat similar to PS, but allow for a more expressive definition of the message space. That is, an *originator* can delegate the signing rights for templates[12] containing fixed and exchangeable (multiple-choice) elements to a *proxy*. The proxy can then choose one of the predefined messages per exchangeable element and issue a signature on such an instantiation on behalf of the originator.

3.2.1 Formal Definitions

Formally, BDS are defined as follows[13] :

$\mathsf{KeyGen}(\kappa, t)$**:** On input of a security parameter κ and an upper bound for the template size t the public parameters pp are generated. We assume pp to be an input to all subsequent algorithms.

$\mathsf{Sign}(\mathcal{T}, \mathsf{dsk_O}, \mathsf{dpk_P})$**:** Given a template $\mathcal{T}$, the secret signing key of the originator $\mathsf{dsk_O}$ and the public verification key of the proxy $\mathsf{dpk_P}$, this algorithm outputs a template signature $\sigma_{\mathcal{T}}$ and a secret template signing key for the proxy $\mathsf{sk_P^{\mathcal{T}}}$.

$\mathsf{Verify_T}(\mathcal{T}, \sigma_{\mathcal{T}}, \mathsf{dpk_O}, \mathsf{dpk_P}, \mathsf{sk_P^{\mathcal{T}}})$**:** Given a template $\mathcal{T}$, a template signature $\sigma_{\mathcal{T}}$, the public verification keys of originator and proxy $(\mathsf{dpk_O}, \mathsf{dpk_P})$ and the template signing key of the proxy $\mathsf{sk_P^{\mathcal{T}}}$, this algorithm checks whether $\sigma_{\mathcal{T}}$ is a valid signature for $\mathcal{T}$ and returns 1 on success and 0 otherwise.

[12] One can think of templates as forms containing fixed and multiple choice fields.
[13] Again, we use a more compact representation of the model, which we adapt from [DHS14].

$\mathsf{Inst}(\mathcal{T}, \sigma_{\mathcal{T}}, \mathcal{M}, \mathsf{dsk_P}, \mathsf{sk_P^{\mathcal{T}}})$**:** On input a template $\mathcal{T}$ with corresponding signature $\sigma_{\mathcal{T}}$, an instance $\mathcal{M} \preceq \mathcal{T}$, as well as the secret template signing key $\mathsf{sk_P^{\mathcal{T}}}$ and the secret signing key of the proxy $\mathsf{dsk_P}$, this algorithm outputs a signature $\sigma_{\mathcal{M}}$ for $\mathcal{M}$.

$\mathsf{Verify_M}(\mathcal{M}, \sigma_{\mathcal{M}}, \mathsf{dpk_O}, \mathsf{dpk_P})$**:** Given an instance $\mathcal{M}$, an instance signature $\sigma_{\mathcal{M}}$ and the public verification keys of originator and proxy $(\mathsf{dpk_O}, \mathsf{dpk_P})$, this algorithm verifies whether $\sigma_{\mathcal{M}}$ is a valid signature on $\mathcal{M}$ and $\mathcal{M} \preceq \mathcal{T}$ (for an unknown $\mathcal{T}$). On success, this algorithm outputs 1 and 0 otherwise.

For security, BDS are required to be correct, unforgeable, immutable and private.

Definition 50 (Unforgeability)**.** *For all PPT adversaries $\mathcal{A}$ there exists a negligible function ϵ such that*

$$\Pr\left[\begin{array}{c}\mathsf{pp} \leftarrow \mathsf{KeyGen}(\kappa, t), (\mathsf{dsk_O}, \mathsf{dpk_O}) \leftarrow \mathsf{DKeyGen}(\kappa), (\mathsf{dsk_P}, \mathsf{dpk_P}) \leftarrow \mathsf{DKeyGen}(\kappa), \\ ((\mathcal{T}^*, \sigma_{\mathcal{T}^*}, \mathsf{sk_P^{\mathcal{T}^*}}), (\mathcal{M}^*, \sigma_{\mathcal{M}^*})) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{T}}, \mathcal{O}_{\mathcal{M}}}(\mathsf{pp}, \mathsf{dpk_O}, \mathsf{dpk_P}) : \\ (\mathsf{Verify_T}(\mathcal{T}^*, \sigma_{\mathcal{T}^*}, \mathsf{dpk_O}, \mathsf{sk_P^{\mathcal{T}^*}}, \mathsf{dpk_P}) = 1 \quad \wedge \quad \mathcal{T}^* \notin \mathcal{Q}_{\mathcal{T}}) \quad \vee \\ (\mathsf{Verify_M}(\mathcal{M}^*, \sigma_{\mathcal{M}^*}, \mathsf{dpk_P}, \mathsf{dpk_O}) = 1 \quad \wedge \quad \mathcal{M}^* \notin \mathcal{Q}_{\mathcal{M}^*})\end{array}\right] \leq \epsilon(\kappa),$$

where $\mathcal{Q}_{\mathcal{T}}$ is the list of queried templates and $\mathcal{Q}_{\mathcal{M}^*}$ is the list of queried instances for template $\mathcal{T}^*$. The adversary $\mathcal{A}$ has access to a template signing oracle $\mathcal{O}_{\mathcal{T}}$ issuing templates signatures for templates of its choice and an instance signing oracle $\mathcal{O}_{\mathcal{M}}$ issuing instance signatures for previously queried templates $\mathcal{T}_i$.

Definition 51 (Immutability)**.** *For all PPT adversaries $\mathcal{A}$ there exists a negligible function ϵ such that*

$$\Pr\left[\begin{array}{c}\mathsf{pp} \leftarrow \mathsf{KeyGen}(\kappa, t), (\mathsf{dsk_O}, \mathsf{dpk_O}) \leftarrow \mathsf{DKeyGen}(\kappa), (\mathsf{dsk_P}, \mathsf{dpk_P}) \leftarrow \mathsf{DKeyGen}(\kappa), \\ ((\mathcal{T}^*, \sigma_{\mathcal{T}^*}, \mathsf{sk_P^{\mathcal{T}^*}}), (\mathcal{M}^*, \sigma_{\mathcal{M}^*})) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{T}}}(\mathsf{pp}, \mathsf{dpk_O}, \mathsf{dpk_P}, \mathsf{dsk_P}) : \\ (\mathsf{Verify_T}(\mathcal{T}^*, \sigma_{\mathcal{T}^*}, \mathsf{dpk_O}, \mathsf{sk_P^{\mathcal{T}^*}}, \mathsf{dpk_P}) = 1 \quad \wedge \quad \mathcal{T}^* \notin \mathcal{Q}_{\mathcal{T}}) \quad \vee \\ (\mathsf{Verify_M}(\mathcal{M}^*, \sigma_{\mathcal{M}^*}, \mathsf{dpk_P}, \mathsf{dpk_O}) = 1 \quad \wedge \quad \mathcal{M}^* \not\preceq \mathcal{T}^*)\end{array}\right] \leq \epsilon(\kappa),$$

where $\mathcal{Q}_{\mathcal{T}}$ is the list of queried templates. The adversary $\mathcal{A}$ has access to a template signing oracle $\mathcal{O}_{\mathcal{T}}$ issuing templates signatures for templates of its choice and returning the respective template secret keys as well as an instance signing oracle $\mathcal{O}_{\mathcal{M}}$ issuing instance signatures for previously queried templates $\mathcal{T}_i$.

Definition 52 (Privacy)**.** *For all PPT adversaries $\mathcal{A}$ there exists a negligible function ϵ such that*

$$\Pr\left[\begin{array}{c}\mathsf{pp} \leftarrow \mathsf{KeyGen}(\kappa, t), (\mathsf{dsk_O}, \mathsf{dpk_O}) \leftarrow \mathsf{DKeyGen}(\kappa), (\mathsf{dsk_P}, \mathsf{dpk_P}) \leftarrow \mathsf{DKeyGen}(\kappa), \\ (\mathcal{T}_0, \mathcal{T}_1, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{T}}, \mathcal{O}_{\mathcal{M}}}(\mathsf{pp}, \mathsf{dpk_O}, \mathsf{dpk_P}), \\ (\mathcal{T}_b, \sigma_{\mathcal{T}_{b'}}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{T}}, \mathcal{O}_{\mathcal{M}}, \mathcal{O}_{\mathcal{M}'}}(\mathsf{state}, \sigma_{\mathcal{T}_0}, \sigma_{\mathcal{T}_1}) : b = b'\end{array}\right] \leq \frac{1}{2} + \epsilon(\kappa),$$

where the adversary in the first phase has access to $\mathcal{O}_{\mathcal{T}}$ and $\mathcal{O}_{\mathcal{M}}$ as in the unforgeability definition and ends this phase by outputting two templates which allow instantiation of $k > 1$ equal messages. Then in the second phase the adversary gets the two templates signatures in a randomly permuted order, has access to $\mathcal{O}_{\mathcal{T}}$ and $\mathcal{O}_{\mathcal{M}}$ as in phase 1 (excluding $\mathcal{T}_0$ and $\mathcal{T}_1$) and can additionally query a modified instantiation oracle $\mathcal{O}_{\mathcal{M}'}$ at most k times for instances $\mathcal{M} \preceq \mathcal{T}_0, \mathcal{T}_1$ and obtains the output of Inst *for both templates in a randomly permuted order.*

3.2.2 Instantiations

There are two known instantiations of BDS, i.e., [HS13a] and [DHS14]. The former uses EUF-CMA digital signatures and a variation of the polynomial commitments in [KZG10] to encode templates and messages, and, thus, reaches succinct signatures. The latter is a black-box construction of BDS from non-interactive attibute- based anonymous credentials.

3.3 Policy-Based Signatures

As already mentioned in Section 2.2, [BF14] also provide an efficient instantiation of a PBS scheme. It basically follows the generic construction presented in Section 2.2, but does not require the encryption scheme, because the used GS proofs directly allows to extract the witnesses. This means that the encrypted witnesses are not required for the extraction in the simulator. Due to the similarity to the generic construction, however, we do not restate the instantiation here and refer the reader to [BF14] for more details.

3.4 Attribute-Based Signatures

Attribute-based signatures (ABS) [MPR08, MPR11] allow users who are issued signing keys for attributes by authorities to produce signatures for any message "tagged" with a predicate that is satisfied by the user's attributes. Thereby, those predicates can be chosen by the user in an ad-hoc fashion and signatures do not reveal anything beyond that the predicate is satisfied, i.e., a signature does not leak which attributes have been used to satisfy the predicate. Consequently, the verification of a signature only tells that the signer who has produced the signature has a signing key to attributes that satisfy the predicate but neither reveals the signers identity nor the set of attributes for which the signing key has been issued.

3.4.1 Formal Definitions

First, we define monotone span programs and note that every linear secret sharing scheme can be derived from a monotone span program computing its access structure [KW93]. We call ℓ the length and t the width of the span program and $\ell + t$ the size of the span program.

Definition 53 (Monotone Span Program [MPR11])**.** *Let $\Upsilon : \{0,1\}^n \to \{0,1\}$ be a monotone boolean function. A monotone span program for Υ over a field $\mathbb{F}$ is an $\ell \times t$ matrix $\mathbf{M}$ with entries in $\mathbb{F}$, along with a labeling function $a : [\ell] \to [n]$ that associates each row in $\mathbf{M}$ with an input variable of Υ, that, for every $(x_1, \ldots, x_n) \in \{0,1\}^n$, satisfies the following:*

$$\Upsilon(x_1, \ldots, x_n) = 1 \iff \exists \vec{v} \in \mathbb{F}^{1\times\ell} : \vec{v}\mathbf{M} = [1, 0, 0, \ldots, 0] \text{ and } (\forall i : x_{a(i)} = 0 \Rightarrow v_i = 0)$$

Intuitively, $\Upsilon(x_1, \ldots, x_n) = 1$ if and only if the rows of $\mathbf{M}$ indexed by $\{i \mid x_{a(i)} = 1\}$ span the vector $[1, 0, 0, \ldots, 0]$.

In the following let $\mathbb{A}$ be the universe of all attributes and let us define a *claim predicate* over $\mathbb{A}$ to be a monotone boolean function whose inputs are associated with attributes in $\mathbb{A}$. One says that a set of attributes $\mathbb{A}' \subseteq \mathbb{A}$ *satisfies* a claim predicate Υ if we have that $\Upsilon(\mathbb{A}') = 1$.

The important parties involved in ABS are a *signature trustee* (TI), who is responsible for the setup (of common parameters used by all parties, i.e., a common reference string), an *attribute-issuing authority* (AIA), who issues signing keys to a set of signers, as well as a set of signers and verifiers.

Definition 54 (Attribute-Based Signatures [MPR11])**.** *An attribute-based signature (ABS) scheme is parametrized by a universe of possible attributes $\mathbb{A}$ and message space $\mathcal{M}$ and consists of the following algorithms:*

$\mathsf{ABS.TSetup}(1^\kappa)$**:** *On input a security parameter κ generate and output a public reference information* tpk *(this algorithm is run by a* signature trustee*).*

$\mathsf{ABS.ASetup}(1^\kappa)$**:** *On input a security parameter κ generate and output a key pair* $(\mathsf{ask}, \mathsf{apk})$ *(this algorithms is run by an* attribute-issuing authority*).*

$\mathsf{ABS.AttrGen}(\mathsf{ask}, \mathbb{A}')$**:** *On input* ask *and $\mathbb{A}' \subseteq \mathbb{A}$ output a signing key* $\mathsf{sk}_{\mathbb{A}'}$*.*

$\mathsf{ABS.Sign}(\mathsf{pk}, \mathsf{sk}_{\mathbb{A}'}, \mathsf{m}, \Upsilon)$**:** *On input* $\mathsf{pk} = (\mathsf{tpk}, \mathsf{apk})$*, signing key* $\mathsf{sk}_{\mathbb{A}'}$*, message $m \in \mathcal{M}$ and a claim predicate Υ with $\Upsilon(\mathbb{A}') = 1$ output a signature σ.*

$\mathsf{ABS.Verify}(\mathsf{pk}, \mathsf{m}, \Upsilon, \sigma)$**:** *On input* $\mathsf{pk} = (\mathsf{tpk}, \mathsf{apk})$*, a message $m \in \mathcal{M}$, a claim predicate Υ and a signature σ output either 1 or 0.*

For correctness one requires that for all $\mathsf{tpk} \leftarrow \mathsf{ABS.TSetup}(1^\kappa)$, all purported apk, all messages $m \in \mathcal{M}$, all attribute sets $\mathbb{A}' \subseteq \mathbb{A}$, all signing keys $\mathsf{sk}_{\mathbb{A}'} \leftarrow \mathsf{ABS.AttrGen}(\mathsf{ask}, \mathbb{A}')$, all claim predicates Υ such that $\Upsilon(\mathbb{A}') = 1$ and all signatures $\sigma \leftarrow \mathsf{ABS.Sign}(\mathsf{pk}, \mathsf{sk}_{\mathbb{A}'}, \mathsf{m}, \Upsilon)$ we have that $\mathsf{ABS.Verify}(\mathsf{pk}, \mathsf{m}, \Upsilon, \sigma) = 1$.

For unforgeability, we first need to define the oracles available to the adversary.

$\mathcal{O}^{\mathsf{ABS.AttrGen}}(\mathsf{ask}, \mathbb{A}')$**:** This works exactly as the algorithm in Definition 54.

$O^{\mathsf{ABS.Sign}}(\mathsf{ask}, m, \Upsilon)$**:** This oracle first runs $\mathsf{sk}_{\mathbb{A}'} \leftarrow \mathsf{ABS.AttrGen}(\mathsf{ask}, \mathbb{A}')$ for any arbitrary $\mathbb{A}'$ that satisfies Υ. Then it outputs the result of $\mathsf{ABS.Sign}(\mathsf{pk}, \mathsf{sk}_{\mathbb{A}'}, \mathsf{m}, \Upsilon)$.

Henceforth, let us denote by $\mathcal{Q}_{\mathsf{Sign}}$ and $\mathcal{Q}_{\mathsf{Gen}}$ the set of tuples queried to the $\mathsf{ABS.Sign}$ and $\mathsf{ABS.AttrGen}$ oracles, respectively.

Definition 55 (Unforgeability [MPR11])**.** *An ABS scheme is called* unforgeable *if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{c} \mathsf{tsp} \leftarrow \mathsf{ABS.TSetup}(1^\kappa), (\mathsf{ask}, \mathsf{apk}) \leftarrow \mathsf{ABS.ASetup}(1^\kappa), \\ \mathcal{O} \leftarrow \{\mathcal{O}^{\mathsf{ABS.AttrGen}}(\mathsf{ask}, \cdot), \mathcal{O}^{\mathsf{ABS.Sign}}(\mathsf{ask}, \cdot, \cdot)\}, (m^*, \Upsilon^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{tpk}, \mathsf{apk})) : \\ (m^*, \Upsilon^*) \notin \mathcal{Q}_{\mathsf{Sign}} \ \wedge \ \mathsf{ABS.Verify}(\mathsf{pk}, m^*, \Upsilon^*, \sigma^*) = 1 \ \wedge \ \Upsilon^*(\mathbb{A}') = 0 \ \forall \mathbb{A}' \in \mathcal{Q}_{\mathsf{Gen}} \end{array}\right] \leq \epsilon(\kappa) \ ,$$

Definition 56 (Perfect Privacy [MPR11]). *An ABS scheme is called* perfectly private *if for all honestly generated* $\mathsf{tpk} \leftarrow \mathsf{ABS.TSetup}(1^\kappa)$*, all purported* apk*, all attribute sets* $\mathbb{A}_1$*,* $\mathbb{A}_2$*, all all signing keys* $\mathsf{sk}_{\mathbb{A}_1} \leftarrow \mathsf{ABS.AttrGen}(\mathsf{ask}, \mathbb{A}_1)$ *and* $\mathsf{sk}_{\mathbb{A}_2} \leftarrow \mathsf{ABS.AttrGen}(\mathsf{ask}, \mathbb{A}_2)$*, all messages* $m \in \mathcal{M}$ *and all claim predicates* Υ *such that* $\Upsilon(\mathbb{A}_1) = \Upsilon(\mathbb{A}_2) = 1$ *the distributions* $\mathsf{ABS.Sign}(\mathsf{pk}, \mathsf{sk}_{\mathbb{A}_1}, \mathsf{m}, \Upsilon)$ *and* $\mathsf{ABS.Sign}(\mathsf{pk}, \mathsf{sk}_{\mathbb{A}_2}, \mathsf{m}, \Upsilon)$ *are equal.*

3.4.2 Instantiations

Generic Framework. Subsequently, we present the generic framework to construct ABS from [MPR11]. Therefore, we have to introduce so called credential bundles (which can be trivially constructed from any secure digital signature scheme).

Definition 57 (Credential Bundle [MPR11]). *A credential bundle for message space* $\mathcal{M}$ *consists of the following algorithms*

$\mathsf{CB.Setup}(1^\kappa)$: *On input a security parameter* κ *generate and output a private public key pair* $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{CB.Gen}(\mathsf{sk}, m_1, \ldots, m_n)$: *On input a signing key* sk *and a set of messages* $\{m_1, \ldots, m_n\} \subseteq \mathcal{M}$ *output a tag* τ *and values* $\sigma_1, \ldots, \sigma_n$.

$\mathsf{CB.Verify}(\mathsf{pk}, m, \tau, \sigma)$: *On input a verification key* pk*, a message* m*, tag* τ *and a value* σ *output either* 1 *or* 0.

Correctness for a credential bundle requires, that for all $(\tau, \sigma_1, \ldots, \sigma_n) \leftarrow \mathsf{CB.Gen}(\mathsf{sk}, m_1, \ldots, m_n)$ we have that $\mathsf{CB.Verify}(\mathsf{pk}, m_i, \tau, \sigma_i) = 1$ for all $i \in [n]$. The security of a credential bundle basically requires that having some bundles, a new bundle can only be obtained from taking a subset of *one* existing bundle, i.e., attributes from several bundles cannot be combined. We refer the reader to [MPR11] for a formal definition of security, which basically is a variant of EUF-CMA security adopted to the notion of bundles.

Now, we present a generic construction based on a credential bundle and non-interactive witness indistinguishability (NIWI) proofs (cf. Section 1.3.5).

Theorem 9 ([MPR11]). *Given a NIWI argument of knowledge system and any secure credential bundle scheme (equivalently, any secure digital signature scheme), then the generic construction presented below is a secure ABS scheme. Furthermore, if the NIWI argument is perfectly hiding, then the ABS scheme is perfectly private.*

Let $\mathbb{A}$ be the universe of attributes and $\hat{\mathbb{A}}$ denote a space of *pseudo-attributes* such that $\mathbb{A} \cap \hat{\mathbb{A}} = \emptyset$. For every message m and claim-predicate Υ one associates a pseudo-attribute $a_{m,\Upsilon} \in \hat{\mathbb{A}}$. In the following let $(\mathsf{CB.Setup}, \mathsf{CB.Gen}, \mathsf{CB.Verify})$ as well as $(\mathsf{NIWI.Setup}, \mathsf{NIWI.Prove}, \mathsf{NIWI.Verify})$ be a secure credential bundle scheme for message space $\mathbb{A} \cup \hat{\mathbb{A}}$ and NIWI proof system respectively.

$\mathsf{ABS.TSetup}(1^\kappa)$**:** On input a security parameter κ run $\mathsf{crs} \leftarrow \mathsf{NIWI.Setup}(1^\kappa)$ as well as $(\mathsf{tpk}', \mathsf{tsk}) \leftarrow \mathsf{CB.Setup}(1^\kappa)$ and publish $\mathsf{tpk} = (\mathsf{crs}, \mathsf{tpk}')$.

$\mathsf{ABS.ASetup}(1^\kappa)$**:** On input a security parameter κ run and output $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{CB.Setup}(1^\kappa)$.

$\mathsf{ABS.AttrGen}(\mathsf{ask}, \mathbb{A}')$**:** On input ask and $\mathbb{A}' \subseteq \mathbb{A}$, check if $\mathbb{A}'$ contains no pseudo-attributes and output $\mathsf{sk}_{\mathbb{A}'} \leftarrow \mathsf{CB.Gen}(\mathsf{ask}, \mathbb{A}')$.

$\mathsf{ABS.Sign}(\mathsf{pk}, \mathsf{sk}_{\mathbb{A}'}, \mathsf{m}, \Upsilon)$**:** On input $\mathsf{pk} = (\mathsf{tpk}, \mathsf{apk})$, signing key $\mathsf{sk}_{\mathbb{A}'}$, message $m \in \mathcal{M}$ and a claim predicate Υ with $\Upsilon(\mathbb{A}') = 1$, parse $\mathsf{sk}_{\mathbb{A}'}$ as $(\tau, \{\sigma_a \mid a \in \mathbb{A}'\})$. Define $\tilde{\Upsilon} := \Upsilon \vee a_{m,\Upsilon}$ where $a_{m,\Upsilon} \in \hat{\mathbb{A}}$ is the pseudo-attribute associated with (m, Υ). Thus, one still has $\tilde{\Upsilon}(\mathbb{A}') = 1$. Let $\{a_1, \ldots, a_n\}$ be the attributes appearing in $\tilde{\Upsilon}$. Let pk_i be apk if a_i is a pseudo-attribute and tpk otherwise. Finally, let $\Phi[\mathsf{pk}, m, \Upsilon]$ denote the following expression:

$$\exists \tau, \sigma_1, \ldots, \sigma_n : \ \tilde{\Upsilon}\left(\{a_i \mid \mathsf{CB.Verify}(\mathsf{pk}_i, a_i, \tau, \sigma_i) = 1\}\right).$$

For each i, set $\hat{\sigma}_i \leftarrow \sigma_{a_i}$ from $\mathsf{sk}_{\mathbb{A}'}$ if it is present and to any arbitrary value otherwise. Compute $\pi \leftarrow \mathsf{NIWI.Prove}(\mathsf{crs}, \Phi[\mathsf{pk}, m, \Upsilon], (\tau, \hat{\sigma}_1, \ldots, \hat{\sigma}_n))$ and output π as the signature.

$\mathsf{ABS.Verify}(\mathsf{pk}, \mathsf{m}, \Upsilon, \pi)$**:** On input $\mathsf{pk} = (\mathsf{tpk}, \mathsf{apk})$, a message $m \in \mathcal{M}$, a claim predicate Υ and a proof π output the result of $\mathsf{NIWI.Verify}(\mathsf{crs}, \Phi[\mathsf{pk}, m, \Upsilon], \pi)$.

In [MPR11], the authors instantiate this generic framework with Groth-Sahai proofs as the NIWI proof system and 1) Boneh-Boyen signatures [BB04, BB08] as well as 2) Waters signatures [Wat05]. Let s be the size of the MSP, then the approach 1) achieves $O(\kappa \cdot s)$ group elements for a signature and using 2) (which avoids to commit to scalars of Boneh-Boyen signatures bitwise) achieves $O(\kappa + s)$ group elements per signature (but has larger, i.e., $O(\kappa)$ group elements in the public key, in contrast to a constant number for Boneh-Boyen signatures).

As the second more efficient approach still yields very large signatures, [MPR11] also presents a third approach which avoids the use of NIWI proof system and yields signatures of size $O(s+2)$. In contrast to the two other approaches which require the t-SDH and either the DLIN or SXDH assumption or only the DLIN or SXDH assumption the latter scheme relies on an interactive assumption and is directly analyzed in the generic group model.

We note that besides the constructions mentioned above, there are some other constructions [GNS10, LK10, LAS+10, HLLR12] (for limited classes of policies) in the literature.

Non Pairing-Based Constructions. Although the generic construction that we have presented before does not rule out constructions without relying on pairings, the only efficient instantations rely on using Groth-Sahai proofs for the NIWI part and thus are inherently pairing based. Nevertheless, we will briefly discuss recently proposed schemes [Her14, Her15] which rely on RSA-type and discrete logarithm related assumptions respectively.

The RSA-type construction [Her14] works in the strong-RSA setting and a secret key for a user actually is a set of texbook RSA signatures on hash values $H(at_i)$ of attributes at_i under a randomly chosen per user key. Then, a signature is a non-interactive zero-knowledge proof of knowledge of an integer e (the per user signature verification key) and at least ℓ e-th roots modulo N of the values $H(at_1), ..., H(at_n)$ (this construction is for threshold policies, i.e., at

least ℓ attributes have to be present to be able to produce a signature, but [Her14] also discuss how to extend this approach to more general policies).

A more recent approach [Her15] constructs ABS based on Schnorr signatures (i.e., non-interactive honest-verifier zero-knowledge proofs of knowledge of discrete logarithms via Fiat Shamir) in the discrete logarithm setting. Let us assume we have a group $\mathbb{G}$ of prime order q generated by g. In this construction, there is a bound L on the number of secret keys that are issued in the system which has to be fixed before the setup and the size of the public parameters linearly depends on L (which makes the approach not attractive for large user bases). The basic idea when having N attributes and $M = N + L$ is to choose for all $i \in [N], j \in [M]$ independently at random secrets $x_{i,j}$ and publish the parameters $Y_{i,j} = g^{x_{i,j}}$. For a user with attributes S, one issues a key as follows: Choose random $\vec{a} = (a_1, \ldots, a_M) \xleftarrow{R} (\mathbb{Z}_q)^M$ and for each attribute at_i in S to compute $s_i = \sum_{j=1}^{M} a_j x_{i,j} \pmod q$ (it must be assured that all the $\vec{a}$ are linearly independent). A signature then basically is a non-interactive honest verifier zero-knowledge proof of knowledge of a subset $S' \subseteq S$ of sufficient size (at least the threshold) such that the user knows $a_j, s_i \neq 0$ with $g^{s_i} = \prod_{j=1}^{N} Y_{i,j}^{a_j}$.

3.4.3 Extensions

After presenting constructions of ABS, we briefly discuss some extensions of ABS in various different directions.

Multiple Attribute Authorities. Already in the initial work [MPR11], ABS have been defined such that it is possible to model different attribute-issuing authorities (but relying on a single signature trustee). Actually, for the generic framework presented above it is very easy to have different attribute-issuing authorities.

Later, Okamoto and Takashima [OT13] considered multi-authority attribute-based signatures which are fully decentralized (DMA-ABS), i.e., there is no longer any trusted central authority required for the setup (like the signature trustee above). For the sake of completeness we note that the scheme proposed in [OT13], which is secure in the random oracle model under DLIN, is (like it is often seen within identity-based cryptography) a signature scheme obtained via Naor's transform [BF01] from a decentralized multi-authority functional encryption (DMA-FE) scheme.

Revoking Anonymity. Recall that ABS do not reveal the identity of the signer but only that the signer possesses a signing key to some unknown set of attributes that satisfy a policy. There have been some works into the direction of revoking the anonymity of signers later on. The most early approach can be found within attribute-based group signatures (ABGS) [Kha07b] which are an attribute-based version of group signatures (cf. Section 3.5.1) and this revocation feature is inherent to group signatures. However, in contrast to ABS, the concept of ABGS does not require to hide the attributes used to satisfy a policy within a signature (and thus do not satisfy the privacy requirement of ABS).

[EHM11] explicitly studied revocation in context of ABS. This means that there is an authority which is able to identify a signer when given a signature and some additional trapdoor information. In [EHM11], a standard model construction which provides revocation is proposed and the RSA-style scheme secure in the random oracle model from [Her14] can also be easily modified to provide revocation.

Revoking Attributes. One fundamental question in context of ABS is the ability to selectively revoke attributes of users after they have been issued signing keys for certain sets of attributes (or to label attributes with validity periods). The first such approach has been studied in context of ABGS [Kha07a]. Here, a revocation table is used to enable the verifiers to identify (if a signer has been revoked or) if any of his attributes have been revoked before accepting a signature.

Very recently, revocation has for the first time been extensively studied in context of ABS [TV15] (the initial work [MPR11] discussed some of these issues in an informal manner). Basically, the authors extend ABS to support attribute expiration as well as key-update mechanisms that allow efficient extension of issued attribute sets. They provide three instantiations, where basically all instantiation are based on the idea to convert a span program Υ that does not take time into account into a span program Υ' that includes requirements for a validity interval of all attributes used to satisfy Υ. More precisely, in the first approach every attribute is assigned a validity interval that is independent of any other validity interval used by any other attribute and is quite inefficient. The two other approaches are based on so called validity attributes, i.e., an attribute that indicates the interval for which the attributes are valid (there may be different validity attributes for one attribute set). Thereby, the second approach uses one common validity interval for all attributes in an attribute set and the third approach partitions the attributes set into subsets, where all attributes in a subset share the same validity interval.

Richer predicates. While nearly all constructions of ABS focus on monotone predicates, recent work considers more general predicates, i.e., an ABS construction supporting non-monotone access structures in the standard model [OT14] and [OT13] propose a generalization (denoted DMA functional signatures) which support non-monotone access structures combined with inner-product relations.

3.5 Related Concepts

Finally, in this section we want to mention some concepts that are somewhat related to functional signatures.

3.5.1 Group Signatures

Group signatures [CvH91, ACJT00, BBS04] allow a group manager to create a group, and every member of this group can then anonymously create signatures on behalf of the group. The

signing capabilities are, thereby, delegated by issuing per-user-specific keys. We observe that this concept is somewhat related to PS, since one could interpret this process as some kind of key-based delegation (also recall the similarity of the additional properties introduced for anonymous proxy signatures [FP08], i.e., traceability and non-frameability, which are security properties of early group signatures which have later been condensed in fewer properties [BMW03, BSZ05]).

3.5.2 Traceable Signatures & Co

For the sake of completeness, we also mention traceable signatures [KTY04, Cho09, ACHO11]. They can be seen as an extension of group signatures which additionally feature a more privacy friendly tracing mechanisms (instead of the opening known from group signatures). That is, 1) the group manager can provide so called tracing trapdoors that allow to trace all signatures of a particular user, while all other signers still remain anonymous, and 2) users can prove authorship of a certain signature.

Below, we also want to mention some other privacy enhancing variants of group signatures. In group signatures with message dependent opening [SEH+12, OSEH13], the opening authority cannot open a signature unless an additional authority (the so called admitter) admits to open signatures for specified messages. Thus, this restricts the power of the opening authority. Public linking of group signatures of users without identifying them has been investigated in [NFW99]. Public tracing of signers who have produced a number of signatures above a certain threshold has been studied in [Wei05]. There are also approaches where the linkability of signatures is only possible for a specified time frame (by fixing the randomness for a certain time) [MCVH12]. Another direction is to put the user in charge of controlling which signatures can be linked (controlled linkability), as it is used in direct anonymous attestation (DAA) [BCC04] and related schemes [BFG+13]. Finally, there are approaches which support so called controllable linkability [HLhC+11, HLC+13, SSU14]. Here, a dedicated entity, the so-called linking authority, can determine whether two given signatures stem from the same unknown signer.

4 Malleable Signatures For Arithmetics

In this section we review malleable signature schemes for arithmetics and in particular linearly homomorphic signatures, homomorphic signatures for polynomial functions, fully homomorphic signatures as well as homomorphic aggregate signatures. Large parts of this section also appear (verbatim) in a recent survey eprint paper [TDB15].

4.1 From Digital to Homomorphic Signature Schemes

Foremost, we need to clarify what we mean by a homomorphic signature scheme. As we are going to use a different notation as the one introduced in Section 1.3.3, we first provide an alternative definition for digital signature schemes. Afterwards, we present definitions for homomorphic signatures.

4.1.1 Digital Signatures

A digital signature scheme is defined over the following sets [GMR88]: the messages space $\mathcal{M}$, the signature space $\mathcal{Y}$, the set of private keys $\mathcal{K}$ and the set of public keys $\mathcal{K}'$.

Definition 58 (Digital Signatures [MVOV96])**.** *A* digital signature scheme *is a tuple of the following probabilistic, polynomial-time algorithms:*

- $\mathrm{Set}(1^\lambda)$. *It takes as input a security parameter λ in unary. It outputs a secret key k and the respective public key k'. The public key determines the space of messages $\mathcal{M}$ and the space of signatures $\mathcal{Y}$.*
- $\mathrm{Sig}(k, m)$. *It takes as input a secret key k and a message $m \in \mathcal{M}$. It outputs a signature $\sigma \in \mathcal{Y}$, which is the signature for the message m signed by means of the secret key k.*
- $\mathrm{Vrf}(k', m, \sigma)$. *It takes as input a public key k', a message $m \in \mathcal{M}$ and a signature $\sigma \in \mathcal{Y}$. It outputs "*ok*" if σ is a valid signature of the message m under the secret key k. It outputs "*bad*" otherwise.*

The verification is a fundamental step to check the authenticity of the signature. Specifically, if k' is the corresponding public key of the secret one k, then

$$\forall m \in \mathcal{M}, \quad \mathrm{Vrf}\ (k', m, \mathrm{Sig}(k, m)) = \mathrm{ok}.$$

The verification is a public procedure. Indeed, *anyone*, using the public key, can verify that the signature correctly belongs to the person who holds the private key.

Definition 59. *A digital signature scheme is* correct *if, for any signature σ produced by* Sig *on message $m \in \mathcal{M}$ and private key k, then* $\mathrm{Vrf}(k', m, \mathrm{Sig}(k, m)) = \mathrm{ok}$.

For the definition of security against adaptive adversaries, i.e., the standard EUF-CMA security notion for digital signature schemes, we refer the reader to Section 1.3.3.

4.1.2 Homomorphic Signatures

The work by Johnson et al. (see [JMSW02]) is the first one which provides a rigorous definition of what a homomorphic signature is. The specifications needed to define a homomorphic signature scheme are the same as for the digital counterpart: the message space $\mathcal{M}$ equipped with an operation, the space of signatures $\mathcal{Y}$ (which in this case is also equipped with an operation) and the space for the secret ($\mathcal{K}$) and the public ($\mathcal{K}'$) keys respectively. Of course, the setup, the signing, and the verifying algorithms are still absolutely necessary. In the following we propose the original definition of [JMSW02].

Definition 60. *Assume we have a signature algorithm* Sig *and a verification algorithm* Vrf*, together with a binary operation "$\cdot$". Then we say that* Sig *is a* homomorphic signature with respect to $\cdot$ *if it comes with an efficient family of binary operations* $*_{k'} : \mathcal{Y} \times \mathcal{Y} \to \mathcal{Y}$ *such that, having the messages* $m, m' \in \mathcal{M}$ *and the signatures* $y, y' \in \mathcal{Y}$ *for which*

$$\mathrm{Vrf}(k', m, y) = \mathrm{ok} = \mathrm{Vrf}(k', m', y')$$

for a public key $k' \in \mathcal{K}'$*, then there is a secret key* $k \in \mathcal{K}$ *such that*

$$y *_{k'} y' = \mathrm{Sig}(k, m \cdot m') \quad \textit{and} \quad \mathrm{Vrf}(k', m \cdot m', y *_{k'} y') = \mathrm{ok}.$$

With respect to the framework for digital signatures presented in Section 4.1.1, for homomorphic signatures there are some new elements to take into account:

- *the set* $\mathcal{F}$ *of admissible functions*: this set defines which are the possible computations over the signed data that the homomorphic signature scheme can support. An element of $\mathcal{F}$ is a function $f : \mathcal{M}^N \to \mathcal{M}$.
- *the integer* N: this value gives the maximum data size, i.e. the maximum number of messages that the admissible function can operate on. Indeed it corresponds to the dimension of the message space where those admissible functions are defined.
- *the index* i: this index doesn't have an intrinsic meaning for the homomorphic scheme. It is just a practical tool to keep track of which of the message of the dataset $(m_1, m_2, \ldots, m_N) \in \mathcal{M}$ we are working on.
- *the tag* τ: this element is necessary to define a strong notion of security also for homomorphic signatures.

A fourth new algorithm is also introduced to the ones already present for common digital signature schemes. That is the algorithm $\mathrm{Eval} : \mathcal{K}' \times \{0,1\}^\lambda \times \mathcal{F} \times \mathcal{Y}^{\mathcal{N}} \to \mathcal{Y}$, which is the core part of a homomorphic signature scheme. Indeed it lets anybody compute on authenticated data, i.e., translating functions on messages into functions on signatures. In the following we denote those signatures to the messages $m_1, m_2, \ldots, m_N$ with $\sigma_1, \sigma_2, \ldots, \sigma_N$. Furthermore, we will denote the tuple $(\sigma_1, \sigma_2, \ldots, \sigma_N)$ with $\vec{\sigma}$.

Note that the other three algorithms now behave slightly different because of the new parameters they have in the input. Modeling on the operative definition for digital signatures and taking into account such differences, we describe the corresponding homomorphic signatures as follows [Fre12].

Definition 61. *A* homomorphic signature scheme *is a tuple of the following PPT algorithms:*

- Set $(1^\lambda, N)$. *It takes as input a security parameter* λ *in unary and an integer* N. *It outputs a secret key* k *and the respective public key* k'. *The public key determines the space of messages* $\mathcal{M}$, *the space of signatures* $\mathcal{Y}$, *and the set* $\mathcal{F}$ *of admissible functions* $f : \mathcal{M}^N \to \mathcal{M}$.
- Sig (k, τ, m, i). *It takes as input a secret key* k, *a tag* $\tau \in \{0,1\}^\lambda$, *a message* $m \in \mathcal{M}$, *and an index* $i \in \{1, 2, \ldots, N\}$. *It outputs a signature* $\sigma \in \mathcal{Y}$, *computed using the secret key* k, *which is the signature for the* i*-th message* m *of the dataset tagged by* τ.
- Vrf (k', τ, m, σ, f). *It takes as input a public key* k', *a tag* $\tau \in \{0,1\}^\lambda$, *a message* $m \in \mathcal{M}$, *a signature* $\sigma \in \mathcal{Y}$, *and a function* $f \in \mathcal{F}$. *It outputs "*ok*" if* σ *is a valid signature for the message* m, *output of the function* f *over the dataset tagged by* τ, *under the public key* k'. *It outputs "*bad*" otherwise.*
- Eval $(k', \tau, f, \vec{\sigma})$. *It takes as input a public key* k', *a tag* $\tau \in \{0,1\}^\lambda$, *a function* $f \in \mathcal{F}$ *and a tuple of signatures* $\vec{\sigma} \in \mathcal{Y}^N$. *It outputs a signature* $\sigma' \in \mathcal{Y}$ *for a function* $f \in \mathcal{F}$ *over the (tuple of) signatures* $\vec{\sigma} \in \mathcal{Y}^N$, *labeled by tag* $\tau \in \{0,1\}^\lambda$.

The definition of *correctness* is now provided.

The index i that the algorithm *Sig* takes as input actually indicates that the third term in input, m, is the i-th message in the list of messages $m_1, m_2, \ldots, m_N$. We can then define the following projector:

$$\pi_i : \mathcal{M}^N \to \mathcal{M} \quad \text{such that} \quad (m_1, m_2, \ldots, m_N) \mapsto m_i,$$

where $\pi_i \in \mathcal{F}, \quad \forall\, i \in [N]$. A signature scheme is said correct if it verifies the projection for any signed message $m \in \mathcal{M}$ and if it also verifies the output $f(m_1, m_2, \ldots, m_N) \in \mathcal{M}$. More precisely, this means that beyond verifying computations on multiple signatures, first of all the single signatures have to be valid by themselves. In the latter case, as we said, the verification algorithm takes as admissible function the projector on a single message (i.e. $Vrf(k', \tau, m, \sigma, \pi_i)$), reducing *Vrf* to the usual verification algorithm of a digital signature (i.e. $Vrf(k', \tau, m, \sigma)$). For this reason, we will denote it with just four parameters in the input, omitting the function π_i.

Definition 62. *A homomorphic signature scheme is* correct *if for each output by the algorithm* $\mathrm{Set}(1^\lambda, N)$ *the following conditions are valid:*

(1) For all $\tau \in \{0,1\}^\lambda$, *for all* $m \in \mathcal{M}$, *if* σ *is the output of* $\mathrm{Sig}(k, \tau, m, i)$, *then*

$$\mathrm{Vrf}(k', \tau, m, \sigma) = \mathrm{ok}.$$

(2) For all $\tau \in \{0,1\}^\lambda$ *and for all the* (m_i, σ_i) *pairs for which* $\mathrm{Vrf}(k', \tau, m_i, \sigma_i) = \mathrm{ok}$*, with* $i \in \{1, 2, \ldots, N\}$*,*

$$\mathrm{Vrf}(k', \tau, f(\vec{m}), \vec{\sigma}, \mathrm{Eval}(k', \tau, f, \vec{\sigma})) = \mathrm{ok},$$

where $\vec{m} := (m_1, m_2, \ldots, m_N)$ *and* $\vec{\sigma} := (\sigma_1, \sigma_2, \ldots, \sigma_N)$*.*

4.1.3 Security of Homomorphic Signatures

The strongest security notion for digital signatures cannot be achieved by the homomorphic ones. That is due to their intrinsic design: as Johnson et al. pointed out in [JMSW02], no homomorphic signature will ever be safe against existential forgeries. Clearly, if the signature is homomorphic with respect to the operation "$*$", by definition, this means that from two signatures on the messages m_1 and m_2, one can directly compute a signature on $m_1 * m_2$. Therefore, the only thing we can require is that no one is able to forge signatures on messages outside $\mathrm{span}_*(m_1, m_2, \ldots, m_N)$.

It seems that the homomorphic signatures' notion of security is correlated to one of the following two facts:

(1) the size of $\mathrm{span}_*(m_1, m_2, \ldots, m_N)$ with respect to the number of elements $m_1, m_2, \ldots, m_N$ itself. What we wish to have is indeed that a big set of messages span into a small space, or at least that a few messages don't have a large span.

(2) the hardness of the decomposition problem inside $\mathrm{span}_*(m_1, m_2, \ldots, m_N)$. We would like indeed that it is difficult to write a message $m \in \mathrm{span}_*(m_1, m_2, \ldots, m_N)$ in terms of $m_1, m_2, \ldots, m_N$.

If the decomposition problem is difficult, then the scheme can be secure even if the first requirement is not satisfied. The contrary also holds: if the scheme doesn't present a large span for a small set of messages, it is still secure even though the decomposition is easy. Since being able to quickly decompose a message may be useful also for the legitimate signer (see [Mol03]), we do not consider the issue of having a hard decomposition. Therefore the definition of security of a homomorphic signature is presented as follows (see [JMSW02]).

We define the *advantage* of an adversary $\mathcal{A}$ as the probability that she outputs a valid signature (m', σ') for some message $m' \notin \mathrm{span}_*(m_1, m_2, \ldots, m_N)$, after adaptively querying signature for messages $m_1, m_2, \ldots, m_N$.

Definition 63. *A homomorphic signature scheme is* (t, q, ε)- secure *against existential forgeries (with respect to* $*$*), if every adversary* $\mathcal{A}$ *making no more than* q *chosen-message queries and running in time at most* t *has advantage at most* ε*.*

4.2 Homomorphic Signature Schemes

In this section we present two types of signature schemes satisfying homomorphic properties. First we provide a description of those schemes having the homomorphic property only. Afterwards, we discuss the homomorphic signature schemes presenting also the aggregate property.

4.2.1 Types of Homomorphic Signature Schemes

The whole set of homomorphic signatures can be divided according to the admissible function each scheme supports. Specifically, we can distinguish:

- linearly homomorphic signatures;
- homomorphic signatures for polynomial functions;
- fully homomorphic signatures.

The signatures are presented in the same order they are listed above. That is, they are discussed with respect to an admissible function which is less and less restrictive. For each section, the differences with respect to the general definition of homomorphic signatures are highlighted. Plus, the evolution from linearly up to fully is shown.

Linearly Homomorphic Signatures

A homomorphic signature scheme is linearly homomorphic used when the signed messages are operated by linear functions.

Due to this specific instantiation, with respect to the general definition of homomorphic signature scheme, there are some differences to point out:

- the messages space $\mathcal{M}$ is the vector space ${\mathbb{F}_p}^N$ of dimension N defined over the finite field $\mathbb{F}_p$, for a prime number p. Such p is an additional output of the setup algorithm *Set*;
- the messages are vectors. More precisely, they are elements $v \in {\mathbb{F}_p}^N$, i.e. $v = (a_1, a_2, \ldots, a_N)$ where $a_i \in \mathbb{F}_p$;
- if we consider the vectors $v_1, v_2, \ldots, v_N$, then the set $\mathcal{F}$ of admissible functions $f \in \mathcal{F}$ are all the possible linear combinations in the $\mathbb{F}_p$-linear span of $v_1, v_2, \ldots, v_N$.

Therefore, the homomorphic property in this context is specified as follows. Given *one* signature *per* message $v_1, v_2, \ldots, v_N$ in ${\mathbb{F}_p}^N$, *anyone* can compute a signature for a vector $v' \in {\mathbb{F}_p}^N$, where:

- $v' := f(\vec{v}) = \sum_{i=1}^{N} c_i v_i$, where $\vec{v} := (v_1, v_2, \ldots, v_N)$;
- $c_1, c_2, \ldots, c_N \in \mathbb{F}_p$.

The definition of linearly homomorphic signature follows (see [BFKW09]).

Definition 64. *A* linearly homomorphic signature scheme *is a tuple of the following probabilistic, polynomial-time algorithms:*

- $\mathrm{Set}(1^\lambda, N)$. *It takes as input a security parameter* λ *in unary and an integer* N. *It outputs a secret key* k, *the respective public key* k' *and a prime number* p. *The public key determines the space of messages* ${\mathbb{F}_p}^N$, *the space of signatures* ${\mathbb{F}_p}^N$, *and the set* $\mathcal{F}$ *of admissible functions* $f : {\mathbb{F}_p}^N \to {\mathbb{F}_p}^N$.
- $\mathrm{Sig}(k, \tau, v, i)$. *It takes as input a secret key* k, *a tag* $\tau \in \{0,1\}^\lambda$, *a vector* $v \in {\mathbb{F}_p}^N$, *and an index* $i \in \{1, 2, \ldots, N\}$. *It outputs a signature* $\sigma \in {\mathbb{F}_p}^N$, *computed using the secret key* k, *which is the signature for the* i*-th message* v *of the dataset tagged by* τ.
- $\mathrm{Vrf}(k', \tau, v, \sigma, f)$. *It takes as input a public key* k', *a tag* $\tau \in \{0,1\}^\lambda$, *a vector* $v \in {\mathbb{F}_p}^N$, *a signature* $\sigma \in {\mathbb{F}_p}^N$, *and a function* $f \in \mathcal{F}$. *It outputs "*ok*" if* σ *is a valid signature for the vector* v, *output of the function* f *over the dataset tagged by* τ, *under the public key* k'. *It outputs "*bad*" otherwise.*
- $\mathrm{Eval}(k', \tau, f, \vec{\sigma})$. *It takes as input a public key* k', *a tag* $\tau \in \{0,1\}^\lambda$, *a function* $f \in \mathcal{F}$, *and a tuple of signatures* $\vec{\sigma} \in {\mathbb{F}_p}^N$. *It outputs a signature* $\sigma' = \sum_{i=1}^N c_i\sigma_i \in {\mathbb{F}_p}^N$ *for a function* $f \in \mathcal{F}$ *over the (tuple of) signatures* $\vec{\sigma} \in {\mathbb{F}_p}^N$, *labeled by tag* $\tau \in \{0,1\}^\lambda$.

For the *correctness*, we refer to Definition 62, where the message $m \in \mathcal{M}$ is instead a vector $v \in {\mathbb{F}_p}^N$ and $f(\vec{m}) = \sum_{i=1}^N c_i v_i$.

Homomorphic Signatures for Polynomial Functions

A homomorphic signature for polynomial functions is a signature scheme that allows for polynomial functions on signed messages. The first of such schemes is proposed in [BF11a], where actually the polynomials are multivariates and of bounded degree. It can be seen as a generalization of linearly homomorphic schemes, since in the linearly homomorphic schemes the set of admissible functions is nothing but a polynomial of degree one. The definition of homomorphic signatures for polynomial functions we give is a generalization of the original one presented in [BF11a]. Though, the definition will take into account a maximum degree of the polynomials, which are multivariate.

As described in [BF11a], the general framework of such signatures is composed of the proper tools to allow for polynomial functions:

- the message space is a finite field $\mathbb{F}_p$, for a prime number p;
- the space of signed messages is the polynomial ring $R := \mathbb{Z}[x]/\langle F(x)\rangle$, for a monic irreducible polynomial $F(x)$ of degree d. Such polynomial is the new output of the algorithm *Set*;
- the set of admissible functions $\mathcal{F} \subset \mathbb{F}_p[x_1, \ldots, x_N]$ for the variables $x_1, \ldots, x_N$, with coefficients in $\{-y, \ldots, y\}$ and degree at most d, where y and d are positive integers.

Therefore, the homomorphic property in this context is specified as follows. Given *one* signature *per* message $m_1, m_2, \ldots, m_N$, anyone can compute a signature for the polynomial $f(\vec{m}) = \sum_{j=1}^{\ell} c_j Y_j(\vec{m})$ where:

- $\vec{m} = (m_1, m_2, \ldots, m_N)$;
- $\ell := \binom{N+d}{d} - 1$;
- $\{Y_j\}_{j=1}^{\ell}$ is the set of non-constant monomials ${x_1}^{e_1}, {x_2}^{e_2} \ldots, {x_N}^{e_N}$ of degree $\sum e_N \leq d$;
- $c_1, c_2, \ldots, c_l$ are coefficients in $\mathbb{F}_p$.

Definition 65. *A* homomorphic signature scheme for polynomial functions *is a tuple of the following probabilistic, polynomial-time algorithms:*

- $\mathrm{Set}(1^\lambda, N)$. *It takes as input a security parameter λ in unary and an integer N. It outputs a secret key k, the respective public key k', a prime number p, and a monic irreducible polynomial $F(x)$ of degree d. The public key determines the space of messages $\mathbb{F}_p$, the space of signatures R, and the set $\mathcal{F} \subset \mathbb{F}_p[x_1, \ldots, x_N]$ of admissible functions, where $y = poly(\lambda)$ and $d = \mathcal{O}(1)$.*
- $\mathrm{Sig}(k, \tau, m, i)$. *It takes as input a secret key k, a tag $\tau \in \{0,1\}^\lambda$, a message $m \in \mathbb{F}_p$, and an index $i \in \{1, 2, \ldots, N\}$. It outputs a signature $\sigma \in R$, computed using the secret key k, which is the signature for the i-th message m of the dataset tagged by τ.*
- $\mathrm{Vrf}(k', \tau, m, \sigma, f)$. *It takes as input a public key k', a tag $\tau \in \{0,1\}^\lambda$, a message $m \in \mathbb{F}_p$, a signature $\sigma \in R$, and a function $f \in \mathcal{F}$. It outputs "*ok*" if σ is the valid signature for the message m, output of the function f over the dataset tagged by τ, under the public key k'. It outputs "*bad*" otherwise.*
- $\mathrm{Eval}(k', \tau, f, \vec{\sigma})$. *It takes as input a public key k', a tag $\tau \in \{0,1\}^\lambda$, a function $f \in \mathcal{F}$, a tuple of signatures $\vec{\sigma} \in R$. It outputs a signature $\sigma' = f(\vec{\sigma}) \in R$ for a function $f \in \mathcal{F}$ over the (tuple of) signatures $\vec{\sigma} \in R$, labeled by tag $\tau \in \{0,1\}^\lambda$.*

For the *correctness*, we refer to Definition 62, where $f(\vec{m}) = \sum_{j=1}^{\ell} c_j Y_j(\vec{m})$.

Remark 4.1. *We observe that in [BF11a], the computation of $f \in \mathbb{F}_p[x_1, x_2, \ldots, x_N]$ over the tuple of signatures $\vec{\sigma}$ is actually performed in two steps. Indeed f is firstly lifted to a function, $\hat{f} \in \mathbb{Z}[x_1, x_2, \ldots, x_N]$ defined as $\hat{f} := \sum_{j=1}^{l} c_j Y_j(x_1, x_2, \ldots, x_N)$, where $c_1, c_2, \ldots, c_N$ are integer coefficients. Then σ' is the output of $\hat{f}(\vec{\sigma})$.*

Fully Homomorphic Signatures

With the fully homomorphic signatures we are not restricted anymore to perform just one group operation over authenticated messages. Now, being allowed to use both $+$ and $\times$ over a field $\mathbb{F}_p$, we can evaluate any function. Such a function is now described by a *circuit* C with a certain size and a certain depth d. We don't propose here the definition of a fully homomorphic signature scheme, since it is almost the same as for a general homomorphic signature (Definition 61). We just discuss the few variations to take into account.

- As we already said, the function is seen as a circuit, which is denoted as $C : \mathcal{M}^N \to \mathcal{M}$. In literature, the maximum size of the dataset is denoted by l instead of N.
- Instead of the set of admissible function, we have the circuit family, which is denoted by $\mathcal{C}$.
- The algorithm *Setup* outputs the private key and the public key, but not the set of admissible functions anymore.
- The notion of correctness remains the same with the remark that the circuit C can also be a projection circuit P_i, i.e. $P(m_1, \ldots, m_N) = m_i$. That means that the correctness must hold for single-message signatures [BFS14].

For the *correctness*, due to the generality of the function that a fully homomorphic scheme supports, we refer directly to Definition 62, where the description of such f covers all types of functions.

4.2.2 Homomorphic Aggregate Signatures

In this section we discuss signature schemes for the multiuser case. In such scenario, we want to obtain a signature which supports the aggregation of different signatures on different messages, signed by different users, each of them with its own private key. This task is fulfilled by the so-called *aggregate signatures*, firstly introduced by Boneh et al. in [BGLS03b]. In case we also need to compute on the authenticated data that we want to aggregate, we need a so-called *homomorphic aggregate signature scheme*.

In the following, we first discuss the definition and the framework of a general aggregate signature. Then we define what a homomorphic aggregate signature is and finally describe the linearly homomorphic aggregate ones.

Aggregate Signatures

An aggregate signature scheme combines multiple signatures into a single one. If we have N different messages and their N respective signatures, thanks to the aggregate signature scheme it is possible to compute a single signature for all the N messages. Such an aggregated signature is as long as the individual ones.

More precisely, suppose that N users $u_1, u_2, \ldots, u_N$ want to obtain a signature σ which is an aggregation of the respective signatures $\sigma_1, \sigma_2, \ldots, \sigma_N$. Each signature σ_i has been obtained by user u_i authenticating message m_i, using its private key k_i. Indeed each u_i has its own private-public key pair (k_i, k_i'). In addition, the scheme is also requested to be incremental. That is, after aggregating N signatures obtaining the signature σ, it is always possible to aggregate a further one σ_{N+1}. This means that we don't have to start the process from the scratch and re-set the computations for $\sigma_1, \sigma_2, \ldots, \sigma_N, \sigma_{N+1}$. Instead, it is sufficient to generate the final signature σ' by aggregating σ_{N+1} and σ.

As for any other signature, also the aggregate one is defined over the messages space $\mathcal{M}$, the

signatures space $\mathcal{Y}$, and the set of secret and public keys $\mathcal{K}$ and $\mathcal{K}'$, respectively. Novelties are instead introduced as regards the algorithms:

- the algorithm *Set* for the preliminary stage, where for each user the secret key (used in the signing process) and the respective public key (used in the verification process) are chosen;
- the algorithm *Sig* takes as input a secret key, a message and in addition an index $i \in \{1, 2, \ldots, N\}$, in order to specify the user u_i together with its secret key k_i and message m_i;
- the algorithm Vrf takes as input a message, a signature, and a string of n public keys $k'_1, k'_2, \ldots, k'_N$ (indicated by $\vec{k'}$), one for each signer;
- a new algorithm is introduced, that is the algorithm Agg_σ which aggregates the signatures $\sigma_1, \sigma_2, \ldots, \sigma_N$, computing the signature σ.

A more formal and precise definition follows.

Definition 66. *An* aggregate signature scheme *is a tuple of the following probabilistic, polynomial-time algorithms:*

- $\mathrm{Set}(1^\lambda)$. *It takes as input a security parameter λ in unary. It outputs a pair (k_i, k'_i) of secret and public keys for each user i. The public keys determine the space of messages $\mathcal{M}$ and the space of signatures $\mathcal{Y}$.*
- $\mathrm{Sig}(k, m, i)$. *It takes as input a secret key k, a message $m \in \mathcal{M}$, and an index $i \in \{1, 2, \ldots, N\}$. It outputs a signature $\sigma \in \mathcal{Y}$, which is the signature for the i-th message m, by means of the i-th secret key k.*
- $\mathrm{Vrf}(\vec{k'}, m, \sigma)$. *It takes as input the public keys' string $\vec{k'}$, a message $m \in \mathcal{M}$ and a signature $\sigma \in \mathcal{Y}$. It outputs "*ok*" if σ is a valid signature for the message m, under the public keys $\vec{k'}$. It outputs "*bad*" otherwise.*
- $\mathrm{Agg}_\sigma(\vec{k'}, \vec{m}, \vec{\sigma})$. *It takes as input a public keys' string $\vec{k'}$, a messages's string $\vec{m} \in \mathcal{M}$, and a signatures' string $\vec{\sigma} \in \mathcal{Y}$. It outputs a signature $\sigma_{agg} \in \mathcal{Y}$, which is the aggregate signature of the signatures in $\vec{\sigma}$ of the messages in $\vec{m}$, under the public keys in $\vec{k'}$, respectively.*

Now we give the definition of *correctness*. Roughly speaking, an aggregate signature scheme is correct if the verification holds for the independent signatures over the single messages and the aggregate signature over the aggregate message. In the first case, the algorithm *Vrf* takes as input just one public key k', that is the one corresponding to the secret key k by which the single message has been signed. Therefore in this situation the algorithm *Vrf* coincides to the usual one defined for a classical digital signature.

Definition 67. *An aggregate signature scheme is* correct *if for each output (k, k') of the algorithm $\mathrm{Set}(1^\lambda)$, the following conditions are valid:*

(1) For all $i \in \{1, 2, \ldots, N\}$*, if* σ *is the output of* $\mathrm{Sig}(k, m, i)$*, then* $\mathrm{Vrf}(k', m, \sigma) = \mathrm{ok}$.

(2) If σ_i *is the output of* $\mathrm{Sig}(k, \tau, m, i)$*, then* $\mathrm{Vrf}(\vec{k'}, \vec{m}, \mathrm{Agg}_\sigma(\vec{k'}, \vec{m}, \vec{\sigma})) = \mathrm{ok}$.

Remark 4.2. *Everything we have said so far also holds for any arbitrary subset* U *of the* N *users, where* $0 < |U| < N$.

Homomorphic Aggregate Signatures

Homomorphic aggregate signatures combine together two properties which at first seem to be incompatible. It fact, as discussed in ([ZYW12]), it is at the same time:

- a signature which combines signatures without operations on messages, produced by different users (*aggregate signature*);
- a signature which combines signatures on messages from the same user using an admissible function (*homomorphic signature*).

However, a signature offering both of the above properties is very desirable.

In the following, we will add the homomorphic property (that is, the possibility to compute on authenticated data) to the aggregate signatures' definition. Some differences have to be taken into account:

- the algorithm *Set* takes as input a security parameter and, in addition, an integer N, which stands for the maximum number of users the scheme can support. Therefore, the parameter N has to be decided a priori and this means that the incremental property of aggregate signatures is lost;
- the algorithm Agg_σ takes as input a public keys' string, a messages' string, and a signatures' string. In addition, it takes as input a tag $\tau \in \{0,1\}^\lambda$ and an admissible function $f \in \mathcal{F}$, since the computation over the messages $m_1, m_2, \ldots, m_N$ reflects also on the corresponding signatures $\sigma_1, \sigma_2, \ldots, \sigma_N$;
- a new algorithm is introduced, that is the algorithm Agg_m. It takes as input a public keys' string, a tag $\tau \in \{0,1\}^\lambda$, a signatures' string, and an admissible function $f \in \mathcal{F}$. It combines the messages $m_1, m_2, \ldots, m_N$ according to f;
- the algorithm *Sig* takes as input a secret key, a message, and an index. In addition, it takes as input a tag $\tau \in \{0,1\}^\lambda$;
- the algorithm *Vrf* takes as input a public keys' string, and a message. In addition, it takes as input an admissible function $f \in \mathcal{F}$.

The following definition formalizes the modifications discussed in a organic and precise way.

Definition 68. *A* homomorphic aggregate signature scheme *is a tuple of the following probabilistic, polynomial-time algorithms:*

- $\text{Set}(1^\lambda, N)$. *It takes as input a security parameter* λ *in unary and an integer* N. *It outputs* N *pairs* (k_i, k_i') *of secret and public keys, one for each user* i. *The public keys determine the space of messages* $\mathcal{M}$, *the space of signatures* $\mathcal{Y}$, *and the set* $\mathcal{F}$ *of admissible functions* $f : \mathcal{M}^N \to \mathcal{M}$.
- $\text{Sig}(k, \tau, m, i)$. *It takes as input a secret key* k, *a tag* $\tau \in \{0,1\}^\lambda$, *a message* $m \in \mathcal{M}$, *and an index* $i \in \{1, 2, \ldots, N\}$. *It outputs a signature* $\sigma \in \mathcal{Y}$, *computed using the* i*-th secret key* k, *which is the signature for the* i*-th message* m.
- $\text{Vrf}(\vec{k'}, \tau, m, \sigma, f)$. *It takes as input a public keys' string* $\vec{k'}$, *a tag* $\tau \in \{0,1\}^\lambda$, *a message* $m \in \mathcal{M}$, *a signature* $\sigma \in \mathcal{Y}$, *and an admissible function* $f \in \mathcal{F}$. *It outputs "*ok*" if* σ *is a valid signature for the message* m, *signed using the public keys* $\vec{k'}$, *output of the function* f *over the dataset tagged by* τ. *It outputs "*bad*" otherwise.*
- $\text{Agg}_m(\vec{k'}, \tau, \vec{m}, f)$. *It takes as input a public keys' string* $\vec{k'}$, *a tag* $\tau \in \{0,1\}^\lambda$, *a messages' string* $\vec{m} \in \mathcal{M}$, *and an admissible function* $f \in \mathcal{F}$. *It outputs a message* $m_{Agg} \in \mathcal{M}$, *which is the aggregate message, output of the function* f *over the string of messages* $\vec{m}$ *in the dataset labeled by tag* τ, *coming from the users with public keys* $\vec{k'}$, *respectively.*
- $\text{Agg}_\sigma(\vec{k'}, \tau, \vec{\sigma}, f)$. *It takes as input a public keys' string* $\vec{k'}$, *a tag* $\tau \in \{0,1\}^\lambda$, *a signatures' string* $\vec{\sigma} \in \mathcal{M}$, *and an admissible function* $f \in \mathcal{F}$. *It outputs a signature* σ_{Agg}, *which is the aggregate signature, output of the function* f *over the signatures* $\vec{\sigma}$ *in the dataset labeled by tag* τ, *coming from the users with public keys* $\vec{k'}$, *respectively.*

The *correctness* definition takes into account the new algorithm Agg_m and the introduction of the homomorphic property. When we verify a single signature, as for the aggregate signature schemes, the algorithm *Vrf* takes as input just one public key and not N of them. Furthermore, as for the homomorphic signature schemes, the admissible function f is meant as a projection from the dataset to the message in question.

Definition 69. *A homomorphic aggregate signature is* correct *if for each output of secret-public key pair* (k, k') *of the algorithm* $\text{Set}(1^\lambda, N)$, *the following conditions are valid:*

(1) For all $\tau \in \{0,1\}^\lambda$, *for all* $i \in \{1, 2, \ldots, N\}$, *if* σ *is the output of* $\text{Sig}(k, \tau, m, i)$, *then* $\text{Vrf}(k', \tau, m, \sigma) = \text{ok}$.

(2) If σ_i *is the output of* $\text{Sig}(k, \tau, m, i)$, *then*

$$\text{Vrf}(\vec{k'}, \tau, \text{Agg}_m(\vec{k'}, \tau, \vec{m}, f), \text{Agg}_\sigma(\vec{k'}, \tau, \vec{\sigma}, f), f) = \text{ok}.$$

Linearly Homomorphic Aggregate Signatures

The homomorphic aggregate signature schemes present in literature so fare are the linearly ones. That is, the computation supported is a linear combination of different messages $m_1, m_2, \ldots, m_N$ coming from different users. Such computation is then reflected on the signatures counterpart. In fact, the final signature σ' joins together the signatures $\sigma_1, \sigma_2, \ldots, \sigma_N$, according to the same linear combinations as used for the messages.

In order to derive the linearly homomorphic aggregate signatures from the homomorphic aggregate ones, there are some changes to take into account:

- the messages space $\mathcal{M}$ and the signatures space $\mathcal{Y}$ are the vector space ${\mathbb{F}_p}^N$ of dimension N defined over the finite field $\mathbb{F}_p$, for a prime number p. Such p is the new output of the algorithm *Set*;
- the messages are vectors $v \in {\mathbb{F}_p}^N$, i.e. $v = (a_1, a_2, \ldots, a_N)$ where $a_i \in \mathbb{F}_p$;
- if the vectors $v_1, v_2, \ldots, v_N$ are a basis for ${\mathbb{F}_p}^N$, then the set of the admissible functions $f \in \mathcal{F}$ are all the possible linear combinations in the $\mathbb{F}_p$-linear span of $v_1, v_2, \ldots, v_N$.

The definition of a linearly homomorphic aggregate signature scheme shapes then on Definition 68, taking into account that the general admissible function $f \in \mathcal{F}$ now has to be seen as $f = \sum_{i=1}^{N} c_i v_i$ for the messages and $f = \sum_{i=1}^{N} c_i \sigma_i$ for the signatures. The same holds for correctness.

Remark 4.3. *A formal definition of linearly homomorphic aggregate signature is provided in [ZYW12] and [Jin14]. Though, let us notice that in literature the linearly homomorphic aggregate signature is called homomorphic aggregate signature. Indeed, the unique examples available so far allow just for linear combinations and therefore "linearly" is omitted. We prefer instead to specify whether we are talking about a general homomorphic aggregate signature or a linearly one. Also because in the future, schemes supporting less restrictive functions might be introduced.*

4.3 Evaluation of Homomorphic Signature Schemes

Together with security, there are many other properties that should be taken into account when evaluating a homomorphic signature scheme. In fact it might be important that a signature generated according to an admissible function is indistinguishable from the original ones. Or it may be that we need a post-quantum signature scheme that it is expected to face quantum computer attacks. In this case we have to make sure that the underlying hardness assumption is not based on the integer factorization or the discrete logarithm problem. Furthermore, there are situations where computational efficiency and succinctness of the signature are important features.

As many of the scheme proposed in the literature do not provide any concrete figures on their efficiency and they usually provide only a very compact analysis with respect to different security levels, we do think that a fair comparison is not possible. Thus, we omit the dimension *computational efficiency* in our comparison.

4.3.1 Complexity Assumptions

Basically, so far there are three kinds of settings in which a homomorphic signature scheme can be instantiated. They are either built on assumptions related to the discrete logarithm problem

in (bilinear) groups, or based on the RSA, or on lattice problems. The first two cases deal with the classical discrete logarithm and integer Factorization problems which are proven to be not resilient against quantum computer attacks. The lattice based problems instead are assumed to provide protection also against such adversaries. For a formal treatment of the required assumption we refer the reader to Section 1.3.2.

4.3.2 Security Definitions

Two operative definitions of what security (specifically, unforgeability) is, are provided in the next two paragraphs. The second paragraph presents the strong and the more recent one, while the first paragraph provides a weaker definition that might be sufficient for some applications.

Boneh et al.'s Security Definition

As we already mentioned, the tag τ is a useful element in the definition of homomorphic signatures. This in fact allows for a secure homomorphic signature scheme, as Boneh et al. formalized in [BFKW09]. Since the game presented there is specific for the linear ones, we describe a more general definition given in [BF11a].

Let us recall that a homomorphic signature scheme is the tuple $\mathcal{S}$= (*Set*, *Sig*, *Vrf*, *Eval*), where λ is the security parameter and N the maximum data set size.

Definition 70. *A homomorphic signature scheme $\mathcal{S}$ is* unforgeable *if for all N and any probabilistic polynomial-time adversary $\mathcal{A}$, in the following game the advantage of $\mathcal{A}$ is negligible with respect to λ:*

- Set*: The challenger obtains the secret-public key pair (k, k') by running* $\mathrm{Set}(1^\lambda, N)$*. The public key k' determines the space of messages $\mathcal{M}$, the space of signatures $\mathcal{Y}$, and the set $\mathcal{F}$ of admissible functions $f : \mathcal{M}^N \to \mathcal{M}$. The challenger gives k' to $\mathcal{A}$.*
- Queries*: Proceeding adaptively, the adversary $\mathcal{A}$ selects a sequence of data sets $\vec{m}_i \in \mathcal{M}^N$. For each $i \in \{1, 2, \ldots, N\}$, the challenger chooses the tag $\tau_i \in \{0,1\}^\lambda$, uniformly at random. The challenger gives τ_i and the signatures σ_{ij}, output of* $\mathrm{Sig}(k, \tau_i, m_{ij}, j)$ *for $j \in \{1, 2, \ldots, N\}$, to $\mathcal{A}$.*
- Output*: $\mathcal{A}$ outputs:*
  - *a tag $\tau^* \in \{0,1\}^\lambda$;*
  - *a message $m^* \in \mathcal{M}$;*
  - *a signature $\sigma^* \in \mathcal{Y}$.*

As discussed in [Fre12], the forgery can be done by $\mathcal{A}$ in two ways: either it produces a fresh signature for data which it hadn't seen before, or, seeing a particular data set, it is able to authenticate an incorrect value of one of its functions. We call the first case "Type I forgery" while the second one "Type II forgery". We refer to [BF11a] for the following formalization.

The adversary $\mathcal{A}$ *wins* if $\mathrm{Vrf}(k', \tau^*, m^*, \sigma*, f) = 1$ and either

- *Type I forgery*: $\tau^* \neq \tau_i$ for all $i \in \{1, 2, \ldots, N\}$.
- *Type II forgery*: $\tau^* = \tau_i$ for a certain i and $m^* \neq f(\vec{m}_i)$.

Remark 4.4. *Without the tag τ, the definition of security would be weaker. Indeed without such a file identifier, the adversary would just be able to query some messages on M and not on an entire data set (see [Zha10]).*

Freeman's Security Definition

In [Fre12], Freeman strengthens the adversary: $\mathcal{A}$ is not restricted to query all the messages belonging to a given data set at one time. Now the attacker can query *one* message at a time, and choosing the following one based on the output of the previous query. Furthermore, it can do this adaptively within each data set and spread the queries among the data sets. In this way the attacker can win in a third way: "Type III forgery". That is, $\mathcal{A}$ might output a triple (m^*, σ^*, f) where σ^* is the signature over the pair (m^*, f) which corresponds to a previously seen data set. Though, the adversary hasn't queried enough messages on that data set in order to shape precisely the behavior of f. For further details we refer to [Fre12].

4.3.3 Privacy

In many practical applications it is necessary to protect derived signatures' privacy. There are three different notions of privacy, according to the level of protection achieved by a scheme.

Let us call $\sigma_1, \sigma_2, \ldots, \sigma_N$ the set of signatures from which a signature σ' for a message m' is derived. A homomorphic signature scheme is said to be *weakly context hiding* if σ' only reveals information about the corresponding message m', but doesn't leak any information about the dataset $m_1, m_2, \ldots, m_N$ of the respective above signatures.

This notion of privacy has been introduced in [ABC+12], together with its stronger version: the *strong context hiding*. Such privacy level is achieved by signature schemes when it is not even possible to see that the signature σ' has been computed as the output of $\sigma_1, \sigma_2, \ldots, \sigma_N$. This privacy level requires the infeasibility of linking the signature σ' to the original ones $\sigma_1, \sigma_2, \ldots, \sigma_q$, even in the case that they are publicly revealed (see [Jin14]).

A further privacy level is introduced in [ALP12]. According to the authors, the definition in [ABC+12] takes only the indistinguishability from honestly generated signatures into account. In fact, there are signature schemes (like the one presented in [AL11]) which satisfy that property even if an attacker generates the signature using an admissible function. Note that the *strong context hiding* doesn't imply unlinkability when the original signatures are chosen by an attacker. In order to address this, they define a new notion of privacy, said *completely context hiding*, which requires (statistical) context hiding on adversarially chosen signatures with private key exposure (see [Jin14]).

For a more formal discussion of these privacy notions we refer the reader to Section 2.4.1.

4.4 State-of-the-Art of Homomorphic Signatures

In this subsection the state-of-the-art with respect to homomorphic signature schemes is presented. Due to the large number and the different properties, they are discussed in separate groups, according to the admissible functions they support. For all the schemes we review the underlying hardness assumption (setting) and investigate parameters of the scheme such as the signature's length. Afterwards, the general security of the scheme is discussed with respect to type of adversary, achieved level of privacy as well as the model used to prove the security.

4.4.1 Linearly Homomorphic Signatures

The first homomorphic signature schemes introduced were the linearly ones. Though the earliest signature we present is the one proposed by Boneh et al. in 2009 in [BFKW09], it is not the first one in literature. Actually, many schemes had been published before ([CJL09], [ZKMH07], [YWRG08], [YCK10]), but they have already been proven to be not practical or even not completely secure ([Wan10], [Zha10], [DCNR11]). Therefore, we do not discuss them in this work. On the other hand, we must highlight the fact that increasing the processing overhead is the common drawback of all these public-key cryptographic primitives (see [LGK+11]). However, in recent works many improvements have been done in this sense.

Subsequently, we group linearly homomorphic signature schemes into ones that come with a security proof in the random oracle model and ones that come with a security proof in the standard model.

Random Oracle Model

Signing a Linear Subspace: Signature Schemes for Network Coding, by Boneh et al. (2009)

Boneh et al.'s work [BFKW09] is the milestone of linearly homomorphic signatures. Indeed it is considered the first one to provide a practical framework for such schemes and the notion of a weak adversary is defined. The scheme proposed is proven secure assuming that the co-CDH problem is hard (see Section 1.3.2). The scheme is claimed to have low communication overhead, because of the independence of both public-key and signature's sizes with respect to the data size (see [Zha10]). Furthermore, a lower bound on signatures' length is provided, which shows that the construction is optimal in the sense of [BFKW09]. Besides being the first practical scheme, it has initially only investigated in context of a weak adversary (see Section 4.3.2). Though, the signature enjoys actually the stronger property and is strongly context hiding (cf. [ALP13]).

Secure Network Coding Over the Integers, by Gennaro et al. (2010)

In 2010, Gennaro et al. presented in [GKKR10] a RSA-based signature scheme (for a definition of the RSA assumption see Section 1.3.2). The security is proven under the same model as in [BFKW09]: this means that the signature is unforgeable against the weak adversary only. The

level of privacy guaranteed by the scheme has not been specified yet.

The signature scheme described in [BF11b] is the first one to authenticate vectors over binary fields. The hardness assumption exploited in order to prove the security of the scheme is k-SIS (see Section 1.3.2). This signature scheme is built on the one presented in [GPV08], where the signatures are short vectors in lattices. Therefore also for the signatures provided by [BF11b], the length is supposed to be quite reasonable. While the scheme is resistant to the weak adversary only, it is weakly context hiding in terms of privacy. Note that in this case the number of the linear combinations that can be authenticated by the scheme is bounded (see Remark 4.5).

Lattice-Based Linearly Homomorphic Signature Scheme over Binary Fields, by Wang et al. (2013)

The work presented in [WHW13] by Wang et al. succeeds the first homomorphic schemes defined over binary fields [BF11b] and relies on a hardness assumption over lattices, i.e., the SIS assumption (cf. Section 1.3.2).

As pointed out by the authors of [WHW13], the scheme is secure against Type I forgery and Type II forgery (cf. Section 4.3.2); thus it is unforgeable only against the weak attacker. Furthermore, the scheme is weakly context hiding, as defined in Section 4.3.3. With respect to [BF11b], we can then conclude that no improvements have been done in terms of security and privacy.

Remark 4.5. *As discussed in [Jin14], there is an open problem which concerns all the current lattice-based homomorphic signature schemes over binary field. That is, they are L-limited, where L is the upper-bound on the maximum number of signatures that can be combined.*

Standard Model

Homomorphic Network Coding Signatures in the Standard Model, by Attrapadung and Libert (2011)

The signature scheme described in [AL11] is the first linearly homomorphic signature scheme proven in the standard model. The scheme is defined over bilinear groups, where two decisional assumptions (Ass. 1 and Ass. 2, see Section 1.3.2) and one computational (Ass. 3, see Section 1.3.2) are adopted to prove unforgeability. In this case, instead of working with prime fields (like for the scheme described in [BFKW09]), the coordinates have to be chosen in $\mathbb{Z}_N$, where N is the composite order of the bilinear groups. The signature is constant in terms of size: it consists of three group elements. Furthermore, like in [BFKW09], the unforgeability of the scheme can only face the weak adversary defined in Section 4.3.2. Therefore protection against re-randomizing signatures by adversaries is not guaranteed and special countermeasures have to be taken. Note that, according to [Fre12], it is possible to modify the proof in the standard model and to face even the strong adversary. Another valuable property is that the scheme is strongly context hiding (see Section 4.3.3).

Computing on Authenticated Data: New Privacy Definitions and Constructions, by Attrapadung et al. (2012)

The same authors of [AL11] proposed one year later a similar scheme in [ALP12]. It also works in a composite order bilinear group setting and requires assumptions are Ass. 1, Ass. 2 , Ass. 4 and Ass. 5, as defined in Section 1.3.2. This scheme is a direct improvement of the previous one described in [AL11]: in fact, while it achieves strongly context hiding privacy, the signature is 33% shorter. Though, it is proven secure against the weak adversary only.

Adaptive Pseudo-Free Groups and Applications, by Catalano et al. (2011)

The scheme proposed by Catalano et al. in [CFW11] employs and extends the notion of pseudo-free groups introduced by Rivest in [Riv04] The signature relies on the strong-RSA assumption (see Section 1.3.2) and actually the authors proved that all known signature schemes built on strong-RSA assumptions in the standard model are instances of their general framework.

As for [AL11], the unforgeability is proven according to the weak adversary. Even in this case, the proof in the standard model can be modified such that it is secure against the strong adversary defined in Section 4.3.2. The privacy notion has still to be clarified.

The signature is an element of $\mathbb{Z}_n^*$, where n is the RSA modulus. However, the linear combinations are performed over the integers. This leads to the fact that the number of admissible linear combinations is bounded, otherwise the the vector coordinates would grow too large.

Efficient Network Coding Signatures in the standard model, by Catalano et al. (2012)

This scheme, described in [CFW12], was presented by the same authors of [CFW11] one year later. It relies on the same hardness assumption (strong-RSA) of the scheme described in [CFW11]. Though, it is an improvement in terms of size, since the signature is composed of only one group element with respect to the signature of [CFW11]. In addition, the number of admissible linear combinations is not bounded anymore like in [CFW11]. Indeed they are computed modulo a certain prime number such that the vector coefficients cannot grow beyond it. However, the scheme can cope with the weak adversary only.

Except for the improvement of [CFW11], in [CFW12] another scheme is proposed. This one is defined over bilinear groups of prime order p and it is proven secure under the q-SDH assumption (see Section 1.3.2). The scheme can achieve short signatures as well: it consists of one group element and one more element belonging to $\mathbb{Z}_p$.

Improved Security for Linearly Homomorphic Signatures: A Generic Framework, by Freeman (2012)

In [Fre12], Freeman transformed ordinary (i.e., non-homomorphic) signatures into homomorphic schemes. Specifically, the author converted four of them and proved their security in the standard model, maintaining the original hardness assumptions they were relying on. That is, the scheme proposed in [Wat05] under the CDH, the scheme proposed in [BB04] under the q-SDH,

the scheme proposed in [GHR99] under the strong-RSA assumption and the scheme proposed in [HW09] under the RSA assumption. In terms of efficiency and signatures' size, the schemes in [CFW12] achieve better performance. On the other hand, the four schemes in [Fre12] are better in terms of security: the unforgeability is proven against the stronger adversary (which is actually defined for the first time in this paper). Furthermore, the signatures are also weakly context hiding: in fact in this work, also the issue of privacy was firstly highlighted.

Remark 4.6. *Another important feature of the scheme in [CFW12] is that the to-be-signed vectors's length is unbounded. This is not true for the two constructions of [Fre12] relying on RSA.*

Efficient Completely Context-Hiding Quotable and Linearly Homomorphic Signatures, by Attrapadung et al. (2013)

The work presented in [ALP13] by Attrapadung et al. is a linearly homomorphic signature scheme relying on the Flex-DH assumption (defined in Section 1.3.2). This assumption was necessary to improve the signature described in [ALP12] (which is only weakly context hiding) in order to make it completely context hiding. In this way the signature is composed only of group elements. However, the scheme can cope with the weak adversary only.

Overview of Linearly Homomorphic Signature Schemes

In Table 2, we summarize the properties of the linearly homomorphic signatures discussed so far.

Scheme	Setting	Sig. size	Adversary	Privacy	Model
[**BFKW09**]	DLP	constant in N	weak	complete	ROM
[**GKKR10**]	RSA	1 group elem.	weak	$\emptyset$	ROM
[**BF11b**]	Lattice	depends on N	weak	weak	ROM
[**WHW13**]	Lattice	constant in N	weak	weak	ROM
[**AL11**]	DLP	3 group elem.	weak	strong	SM
[**ALP12**]	DLP	$33\% \leq$ [AL11]	*weak*	strong	SM
[**CFW11**]	RSA	2 group elem.	weak	$\emptyset$	SM
[**CFW12**]	RSA	$\leq$ [CFW11]	weak	$\emptyset$	SM
[**Fre12**]	DLP/RSA	$\geq$ [CFW12]	strong	weak	SM
[**ALP13**]	DLP	16 group elem.	*weak*	complete	SM

Table 2: Linearly Homomorphic Signature Schemes

4.4.2 Homomorphic Signature Schemes for Polynomial Functions

Signature schemes for polynomial functions extend the class of admissible signatures from linear functions to polynomial functions of some fixed degree greater than one.

Homomorphic Signatures for Polynomial Functions, by Boneh and Freeman (2011)

This signature scheme proposed in [BF11a] is the first homomorphic signature supporting evaluation of multivariate, bounded-degree polynomials on authenticated data. It relies on the SIS

hardness (see Section 1.3.2) assumption and is defined over ideal lattices. The signature size depends logarithmically on the data size. On the other hand, the scheme can cope only with the weak adversary. The security is proven in the random oracle model. The level of privacy achieved by the signature scheme still needs to be clarified.

Homomorphic Signatures for Polynomial Functions with Shorter Signatures, by Hiromasa et al. (2013)

Hiromasa et al. presented in [HMO13] a signature scheme which improves the one proposed in [BF11a] in terms of the signature's size. On the other hand, the secret key is longer. Regarding all the other properties and parameters, the signature scheme is proven secure in the random oracle model, using SIS as the underlying hardness assumption. Privacy is still a non specified property.

Homomorphic Signatures with Efficient Verification for Polynomial Functions, by Catalano et al. (2014)

The work presented in [CFW14] outperformes both the above signatures (i.e., the ones in [BF11a, HMO13]). However, the scheme in [CFW14] relies on the k-Augmented Power Multilinear Diffie-Hellman Problem (k-APMDHP) introduced in their paper. The scheme relies on multilinear maps and it is still a work-in-progress to define a practical and efficient one (see [CLT15]). On the other hand, the size of the signature, the public key and the secret key increases of a factor d, where d is the maximum degree of the polynomial supported. This drawback is the expense to be paid in order to achieve a better security level than the scheme proposed in [BF11a]. In fact, the signature scheme described in [CFW14] provides two substantial improvements:

(1) the scheme doesn't rely on random oracles anymore and comes with a security proof in the standard model;

(2) the adversary is not assumed to query signatures on messages in a given data set all at once. That is: the scheme is proven to be secure also in the presence of a strong adversary.

However, the open problem stated by the authors in [BF11a] of building a homomorphic signature providing privacy has still not been solved. Indeed none of the three signature schemes supporting polynomial functions is at least weakly-context hiding.

Overview of Homomorphic Signatures for Polynomial Functions

In Table 3, we briefly present the properties of schemes supporting polynomial functions discussed so far.

Scheme	Setting	Sig. size	Adversary	Privacy	Model
[BF11a]	Lattice	$\mathcal{O}(d \log N)$	weak	$\emptyset$	ROM
[HMO13]	Lattice	$\leq$ [BF11a]	weak	$\emptyset$	*ROM*
[CFW14]	DLP	$\mathcal{O}(d^3 + d^2 \log N)$	strong	$\emptyset$	SM

Table 3: Homomorphic Signature Schemes for Polynomial Functions

4.4.3 Fully Homomorphic Signature Schemes

Subsequently, we present homomorphic signature schemes where the class of admissible functions is even larger and ideally not restricted any more.

Leveled Fully Homomorphic Signatures from Standard Lattices, by Gorbunov et al. (2014)

In [GVW14], the first fully homomorphic signature scheme is proposed. That means that the admissible function can be an arbitrary one. The function is described as a circuit whose depth we denote by d. This scheme relies on the hardness of the SIS assumption (cf. Section 1.3.2) and can evaluate arbitrary circuits over signed data. The signature scheme can be proven secure either in the random oracle model or in the standard model. In the former case, we end with short public parameters, while in the latter case they are even longer than the total size of the dataset.

Note that, in both cases, instead, the signature's size is independent of the data size and of the circuit size. Though, this doesn't mean that the signature is short: it is indeed dependent of the depth d of the circuit, which is an *a-priori* fixed parameter. Therefore, even though we can in principle perform any kind of transformation on the authenticated data, this is done at the expense of having a larger and larger signature.

The adversary these schemes can cope with is the weak one. A particular technique is available [BB11] to convert the schemes so that the scheme is secure against the strong adversary. Though, this is possible at the expense of ending with high inefficiency (see [BFS14]): it would then be possible to sign only few and short messages. Such restriction is a devolution regarding the practicability of the schemes in real life. Furthermore, they are claimed not to lack information about the original data beyond the outcome of the transformation itself. Therefore, according to the terminology adopted in this work, weakly context hiding privacy is provided.

Adaptively Secure Fully Homomorphic Signatures Based on Lattices, by Boyen et al. (2014)

In [GVW14] the authors also propose a fully homomorphic signature scheme based on lattices and can be seen as a concurrent work to [GVW14]. Indeed some improvements have been done, though introducing some problems not present in the aforementioned paper. Specifically, the scheme cannot sign arbitrary circuits any more: rather the ones with poly-logarithmic depth or the ones with polynomial depth. In the first case the hardness assumption is the SIS, while in the second case the scheme relies on the sub-exponential SIS. An important improvement of this work is that the scheme can cope with the strong adversary and this is proven in the

standard model. Unfortunately none of the possible level of privacy is achieved, making the protocol not applicable for many real-life applications.

Overview of Fully Homomorphic Signature Schemes

In Table 4, we review the properties of the above schemes.

Scheme	Setting	Sig. size	Adversary	Privacy	Model	Note
[GVW14]	Lattice	poly in depth	weak	weak	SM	large param.
[GVW14]	Lattice	poly in depth	weak	weak	ROM	short param.
[BFS14]	Lattice	not specified	strong	none	SM	poly-log depth

Table 4: Fully Homomorphic Signature Schemes

4.4.4 Homomorphic Aggregate Signatures and Multiple Users Case

In many practical applications it might be necessary to aggregate multiple signatures on messages, produced even by different users. Linearly homomorphic signatures are not required to provide such a property and they do not consider the case of multiple distinct signers.

Among the digital signature schemes, the aggregate ones (see [BGLS03a]) provide such properties. They are designed to "aggregate" N different signatures (each of them has to be on a distinct message) from N different signers by generating a single signature for them. Each message-signature pair comes with an index $i \in [N]$ corresponding to the user i. The resulting aggregate signature (together with the N messages) will then prove to the verifier that each of the N users has signed one (and only one) of the N original messages.

Introducing the homomorphic property into an aggregate scheme is a quite improvement and up to our knowledge, there are only two such schemes [ZYW12, Jin14]. They both support linear operations over binary fields and rely on lattices. Therefore the hardness assumptions leading to unforgeability are supposed to face even quantum computers' attacks. Specifically, [ZYW12] relies on the ISIS assumption (see Section 1.3.2), while [Jin14] relies on SIS. Furthermore, while in [ZYW12] the signature's length is *as long as* that of each original signatures, in [Jin14] the signature is exactly *two times* longer. Instead, the public key length is the same for both schemes.

Both of the signature schemes are proven unforgeable against the strong adversary. Furthermore, the signature in [Jin14] provides also weakly context hiding privacy. Note that this scheme is a variant of the linearly homomorphic signature introduced in [BF11b], where the same privacy property holds and also apply to the multi-users case. Both schemes are proven in the random oracle model.

In short, because of the improvements in terms of signature's size, efficiency and privacy, the signature by [Jin14] is more desirable than the one presented in [ZYW12].

Having the aggregate property is not a necessary condition for a homomorphic signature to deal with different signatures generated by different users. Indeed earlier works presented homomorphic signatures for the multi-users case ([CV10], [YYLF12]). In both proposals, the length of the signatures is constant in the number of users N.

The properties of the two existing linearly homomorphic aggregate signatures discussed are reported in Table 5.

Scheme	Setting	Sig. size	Adversary	Privacy	Model
[**ZYW12**]	Lattice	constant in N	*strong*	$\emptyset$	ROM
[**Jin14**]	Lattice	constant in N	*strong*	weak	ROM

Table 5: Linearly Homomorphic Aggregate Signature Schemes

5 Malleable Signatures For Editing

In this section our focus is on a class of malleable signatures that can be facilitated to allow edits of signed messages on a document level, where the most common approach is to view the message as an (ordered) set of blocks.

In the following, we will present the state-of-the-art in terms of the security properties that can be achieved for various forms of malleable signatures. We discuss append-only signature schemes (AOS) in Section 5.2, sanitizable signature schemes (SSS) in Section 5.3, and redactable signature schemes (RSS) in Section 5.4. Thereby, Section 5.3 (with exception of Section 5.3.3 on ESSS) and Section 5.4 are based upon material from the not yet published PhD-thesis of Henrich C. Pöhls [Pöhed].

Append-only signatures strictly speaking do not allow to "edit" certain blocks (or submessages) of some signed message. However, they still serve a somewhat related purpose. That is, instead of removing certain message blocks, they allow to publicly append message blocks to some signed message and to update the signature correspondingly.

Apart from appending, we further consider two other operations called redaction, i.e., removing data, and sanitisation, i.e,. a controlled modification of data by alteration. See Fig. 3 for some examples with and without visual traces of an edit.

Redactions are best described as blackening certain blocks from a document. A redaction could actually remove blocks without traces or leave blackened blocks, i.e., indicators that a redaction has taken place (they may or may not indicate their position or their length relative to other blocks of the message). We emphasize that for applications which require privacy, restoring removed or blackened parts of a document must be infeasible.

Sanitizations of a document allows modifications such as required in the context of pseudonymization and anonymization (cf. [PH09] for the terminology), e.g., substituting real-names with pseudonyms. Hence, the possible modifications for a sanitization are not just removal of information (as with redaction), but to generalise, disguise or even fully replace the information from blocks by replacement. Sanitization could also be used to provide a mechanism to update signed data after signature creation, e.g., re-compute a result with additional data.

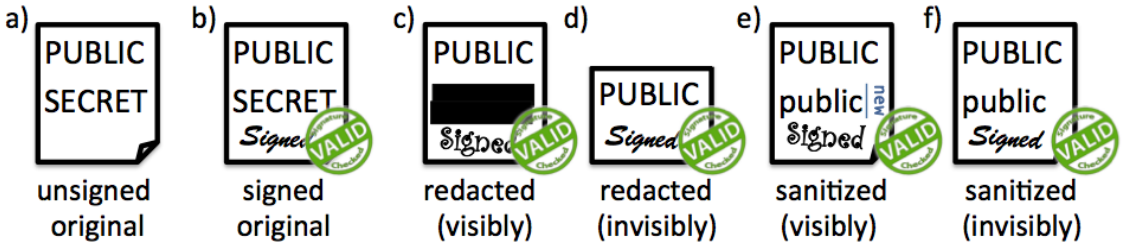

Figure 3: Different types of modification, if the modification of 'SECRET' is authorised the signature still verifies as valid: (a) original (b) signed original (c) visibly redacted + visible position of redaction (d) invisibly redacted (e) visibly sanitized (f) invisibly sanitized [Pöhed]

5.1 Preliminaries and Notation

We assume that there is a decomposition function $\mathsf{decompose}$ that splits a message[14] m into ℓ blocks, for some $\ell \in \mathbb{N}$, where for each $i \in [\ell]$ we have that block $m[i] \in \{0,1\}^*$. Thereby, we use the following conventions:

- With $|m|$ we denote the number of blocks, i.e., $|m| = \ell$.
- We assume that the blocks are not allowed to overlap.
- A block is the smallest unit of data that can be subjected to a modification operation.
- We make no assumption about what is contained in a block, e.g., we do not assume that each block captures a single semantic concept, or that each sentence in a text document is one block.
- Without loss of generality, we assume that all blocks of m are of equal length, i.e., for all $i \in [\ell]$ we have that $m[i] \in \{0,1\}^k$ for some fixed $k \in \mathbb{N}$. This does not impose any restriction, but makes the presentation easier.

In our subsequent presentation a document will always be treated as an ordered set (sequence) of blocks. However, schemes considering tree- [BBD+10, SPB+12b], or set-like structure [JMSW02, PS14] have also been considered and can be found in the literature.

5.2 Append-Only Signatures

While append-only signatures do not allow to "edit" blocks of some signed message, they still serve for a somewhat related purpose. That is, instead of removing certain message blocks, they allow to publicly append message blocks to some signed message and to update the signature correspondingly. Append-only signatures (AOS) were first introduced in [KMPR05]. We recall their formal definitions and the security model below:

Definition 71 (AOS Scheme)**.** *An* AOS *is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{Append}, \mathsf{Vfy})$*, which are defined as follows*

$\mathsf{AOS.Setup}(1^\kappa)$**:** *On input of a security parameter* κ*, this algorithm generates and returns a key pair* $(\mathsf{sk}_{\mathsf{AOS}}, \mathsf{pk}_{\mathsf{AOS}})$*, where* $\mathsf{sk}_{\mathsf{AOS}}$ *is the signature on the empty message.*

$\mathsf{AOS.Append}(\mathsf{pk}_{\mathsf{AOS}}, \sigma_{n-1}, m_n)$**:** *On input of a public key, a signature on a message* $(m_1, \ldots, m_{n-1})$ *and a message* m_n*, this algorithm computes and outputs a signature* σ_n *on the message* $(m_1, \ldots, m_n)$*.*

$\mathsf{AOS.Vfy}(\mathsf{pk}_{\mathsf{AOS}}, M, \sigma)$**:** *On input of a public key* $\mathsf{pk}_{\mathsf{AOS}}$*, a message* $M = (m_1, \ldots, m_n)$ *and a signature* σ*, this algorithm returns* 1 *if* σ *is a valid signature for* M *under* $\mathsf{pk}_{\mathsf{AOS}}$ *and* 0 *otherwise.*

[14] 'message' can be used interchangeably with 'data' or 'document' in this context

For security, AOS are required to be correct and unforgeable under chosen message attacks. While correctness straight-forwardly requires that the verify algorithm always outputs 1 for all honestly generated keys and signatures, unforgeability under chosen message attacks for AOS is defined as follows:

Definition 72 (AOS-UF-CMA)**.** *An* AOS *is called unforgeable under chosen message attacks, if*

$$\left[\begin{array}{c}(\mathsf{sk}_{\mathsf{AOS}},\mathsf{pk}_{\mathsf{AOS}})\leftarrow\mathsf{AOS.Setup}(1^\kappa),(M^*,\sigma^*)\leftarrow\mathcal{A}^{\mathcal{O}^{\mathsf{Sign}}(\mathsf{pk}_{\mathsf{AOS}},\cdot)}(\mathsf{pk}_{\mathsf{AOS}}):\\ \mathsf{AOS.Vfy}(\mathsf{pk}_{\mathsf{AOS}},M^*,\sigma^*)=1\ \wedge\ \nexists\ M\in Q^{\mathsf{Sign}}:M\sqsubseteq M^*\end{array}\right]\leq\epsilon(\kappa),$$

where $\mathcal{O}^{\mathsf{Sign}}(\mathsf{pk}_{\mathsf{AOS}},M)$ *is an oracle that returns signatures* σ *on messages* $M=(m_1,\ldots,m_n)$*,* Q^{Sign} *is used to keep track of the oracle queries, and* $M\sqsubseteq M^*$ *denotes that* M *is a prefix of* M^**.*

5.2.1 Constructions

In [KMPR05], various constructions of AOS are provided. Subsequently, we briefly discuss their underlying principles:

Certificate-Based AOS**.** The idea behind this scheme is to sign pairs of messages and public keys using a conventional signature scheme. More precisely, a signature resembles a chain

$$((m_1,\mathsf{pk}_2),\sigma_1),\ldots,(m_n,\mathsf{pk}_{n+1}),\sigma_n),\mathsf{sk}_{n+1}),$$

where σ_i denotes a conventional signature on (m_i,pk_{i+1}). To append a message m_{i+1} to the signature above, one generates a key pair $(\mathsf{sk}_{n+2},\mathsf{pk}_{n+2})$ and extends the chain to

$$((m_1,\mathsf{pk}_2),\sigma_1),\ldots,(m_n,\mathsf{pk}_{n+1}),\sigma_n),((m_{n+1},\mathsf{pk}_{n+2}),\sigma_{n+1}),\mathsf{sk}_{n+2}).$$

In [KMPR05], the authors additionally proposes to reduce the signature size of the construction above by using sequential aggregate signatures [LMRS04], a primitive that allows to sequentially aggregate signatures to a single signature. However, in this construction it is still required to store n public keys of the sequential aggregate signature scheme.

Hash Tree-Based AOS**.** The basic approach of the hash tree-based construction in [KMPR05] is to use a pseudorandom generator (PRG) to assign random values to the leaves of a binary tree of depth d (i.e., to apply the GMM construction [GGM86] of a PRF from a length doubling PRG). The used seed represents the secret key, and each leave of the tree represents a message $M_i=(m_j)_{j=1}^k$, where $m_j\in\{0,1\}$ and $k\leq d$. Once this tree is built, one uses a hash function to compute the Merkle hash for this tree[15], which is used as public key. Verification amounts to recomputing the root hash, when given the message and corresponding authentication path[16].

[15] That is, the hash value at the root of the tree, which is obtained by recursively computing the pairwise hash of neighbouring nodes.

[16] The hash values of all siblings of the respective node on the path to the root.

Relationship of AOS and HIBE. In [KMPR05], it is shown that hierarchical identity based signature (HIBS) schemes are equivalent to AOS. They show the equivalence based on black-box constructions in both directions. To obtain shorter signatures sizes (square root of the message length), they propose a concrete AOS scheme, based on the BBG HIBE [BBG05] (as HIBS schemes are implied by HIBE schemes). For more details on this construction we refer the reader to [KMPR05].

5.2.2 Extensions

History-Hiding AOS. In [BBW07] the authors require an additional property from AOS which they call *history-hiding*. Informally, this means that one considers sets of messages instead of sequences, and it is required that no PPT adversary can efficiently decide in which order messages were appended. Their construction builds upon symmetric bilinear pairings $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ over groups of prime order p and a hash function $H : \{0,1\}^* \to \mathbb{G}$. More precisely, in their scheme they have $(\mathsf{sk}_{\mathsf{AOS}}, \mathsf{pk}_{\mathsf{AOS}}) \leftarrow (g^\alpha, (g, e(g,g)^\alpha))$, where $\alpha \xleftarrow{R} \mathbb{Z}_p$ and g generates $\mathbb{G}$. A signature on a message $M = (m_1, \ldots, m_n)$ is

$$\sigma = (g^\alpha \cdot \prod_{i=1}^{n} H(m_i)^{r_i}, \bigcup_{i=1}^{n} \{(m_i, r_i)\}),$$

which can be easily verified using the bilinear pairing e, σ and $\mathsf{pk}_{\mathsf{AOS}}$. Finally, we note that quite recently history hiding AOS [LJYP15] that fit in the P-homomorphic signature framework (cf. Section 2.4) were proposed.

Bounded Vector Signatures. Bounded vector signatures (BVS) [WCR11] target a slightly different set of malleable operations. That is, they allow to sign vectors, where signatures are malleable with respect to increasing the values of the entries in the signed vectors up to some predefined maximum value. The authors informally discuss how BVS can be used to obtain signatures on sets and multisets (somewhat similar to history-hiding AOS). Basically, their approach is to assign each possible element in the (multi-)set to a dimension in the vector and to use the vector entries to count the amount of the respective elements in the (multi-)set (for sets, this amount can either be 0 or 1, whereas this value can be larger for multisets).

5.3 Sanitizable Signatures

With sanitizable signatures, a signer can issue signatures on messages consisting of fixed and modifiable blocks. While the fixed blocks cannot be changed without invalidating the signature, modifiable blocks can later be changed (sanitized) by some dedicated party (the sanitizer) without invalidating the signature. With sanitizable signatures, replacements for modifiable (admissible) message blocks can be chosen arbitrarily by the sanitizer.

Sanitizable signatures were firstly introduced in [ACdMT05] and rigorously formalised in [BFF+09]. Subsequently, we present the formalization of [BFF+09].

5.3.1 Formal Definition

A sanitizable signature scheme consists of the subsequent set of efficient (PPT) algorithms. As the exact instantiation of these algorithms differs strongly between the various existing schemes, this section describes each algorithm on the level of its input and output. The entity denoted by signer is always the initial signer of a given message m and sanitizer denotes the entity that is authorised to sanitize the signed message [17].

Definition 73 (Sanitizable Signature Scheme)**.** *SSS consists of at least seven probabilistic polynomial time (PPT) algorithms* $(\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$*, such that:*

Key Generation.

There are two key generation algorithms, one for the signer and one for the sanitizer. Both create a pair of keys consisting of a private key and the corresponding public key:

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

The security parameter is denoted as κ*.*

Signing.

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

$$(m, \sigma) \leftarrow \mathsf{Sign}(m, \mathsf{sk}_{sig}, \mathsf{pk}_{san}, ADM).$$

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

Sanitizing.

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

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

[17] Note, this section uses an array-like notation to index the parts of a message for SSSs without loss of generality. How a structured, unstructured or partly structured message is decomposed into parts must be described by each SSS instantiation individually.

Verification.

The Verify *algorithm outputs a decision as a bit* $d \in \{1, 0\}$ *indicating the validity of the signature* σ *for the message* m *with respect to the public keys* pk_{sig} *and* pk_{san}*.*

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

Proof.

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

$$\pi \leftarrow \mathit{Proof}(\mathsf{sk}_{sig}, m, \sigma, (m_1, \sigma_1), \ldots, (m_k, \sigma_k), \mathsf{pk}_{san}).$$

Judge.

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

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

5.3.2 Security Properties

In this section we give an overview of the most common security properties for SSS. We restate them in a harmonised notation, adopting them by the latest results.

For sanitizable signatures the mandatory security properties are correctness, unforgeability, immutability, transparency, privacy and accountability (sanitizer-accountability and signer-accountability). Other relevant properties that can be valuable in several scenarios are strong privacy [dMPPS14], (strong) unlinkability [BFLS10, BPS13] as well as non-interactive (public) accountability [BPS12].[18]

Correctness of SSS

SSS are assumed to work correctly. That is, the formally defined correctness properties *signing correctness, sanitizing correctness* and *proof correctness* as presented in [BFF+09, BPS12] hold. For readability these are restated in the harmonised notation in the following:

[18] Note that public accountability prohibits transparency. Yet, as detailed in [BPS12], public accountability can be useful in various applications which do not necessarily require transparency.

Definition 74 (Signing correctness for SSS [BFF+09])**.** *All genuinely signed messages* (m, σ) *are accepted as valid by* Verify*.*

For any security parameter κ*, any key pair* $(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathit{KGen}_{sig}(1^\kappa)$*, any key pair* $(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathit{KGen}_{san}(1^\kappa)$*, any message* $m \in \{0,1\}^*$*, any* ADM *and any* $\sigma \leftarrow \mathit{Sign}(m, \mathsf{sk}_{sig}, \mathsf{pk}_{san}, ADM)$ *we have*

$$\mathit{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 1.$$

Definition 75 (Sanitizing correctness for SSS [BFF+09])**.** *All genuinely sanitized messages* (m, σ) *are accepted as valid by* Verify*.*

For any security parameter κ*, any key pair* $(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathit{KGen}_{sig}(1^\kappa)$*, any key pair* $(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathit{KGen}_{san}(1^\kappa)$*, any message* $m \in \{0,1\}^*$*, any* σ *with* $\mathit{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 1$*, any* MOD *matching* ADM *from* σ*, and any pair* $(m', \sigma') \leftarrow \mathit{Sanit}(m, MOD, \sigma, \mathsf{pk}_{sig}, \mathsf{sk}_{san})$ *we require*

$$\mathit{Verify}(m', \sigma', \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 1.$$

Definition 76 (Proof correctness for SSS [BFF+09])**.** *Every genuinely created proof by the signer leads the* `Judge` *to decide in the signer's favour.*

For any security parameter κ*, any key pair* $(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathit{KGen}_{sig}(1^\kappa)$*, any key pair* $(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathit{KGen}_{san}(1^\kappa)$*, any message* $m \in \{0,1\}^*$*, any signature* σ *any* MOD *matching* ADM *from* σ*, any* $(m', \sigma') \leftarrow \mathit{Sanit}(m, MOD, \sigma, \mathsf{pk}_{sig}, \mathsf{sk}_{san})$ *with* $\mathit{Verify}(m\prime, \sigma\prime, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 1$*, and any (polynomially many)* $m_1, \ldots, m_q$ *and* $ADM_1, \ldots, ADM_q$ *with* $\sigma_i \leftarrow \mathit{Sign}(m_i, \mathsf{sk}_{sig}, \mathsf{pk}_{san}, ADM_i)$ *and* $(m, \sigma) = (m_i, \sigma_i)$ *for some* i*, any* $\pi \leftarrow \mathit{Proof}(\mathsf{sk}_{sig}, m', \sigma', m_1, \sigma_1, \ldots, m_q, \sigma_q, \mathsf{pk}_{san})$ *we require:*

$$\mathit{Judge}(m', \sigma', \mathsf{pk}_{sig}, \mathsf{pk}_{san}, \pi) = \mathtt{San}.$$

As mentioned before, ADM is required to always be correctly recoverable from a valid message-signature pair (m, σ) [BFF+09]. Furthermore, [GQZ11] additionally require that unauthorized modifications of ADM count as forgeries (see also [SPPdM12b]). These have been incorporated in the following security property definitions.

Definition 77 (ADM recovery correctness)**.** *Every genuinely created message-signature pair* (m, σ) *allows recovering* ADM *correctly.*

Unforgeability for SSS

Unforgeability guarantees that a valid signature on a message can only be generated by the signer as it requires knowledge of sk_{sig}, or by a sanitizer, requiring knowledge of sk_{san}. Note that valid sanitizations are excluded from being forgeries.

Definition 78 (Unforgeability for SSS [BFF+09, GQZ11])**.** *A* SSS *is* unforgeable*, if for any efficient (PPT) adversary* $\mathcal{A}$ *the probability that the game depicted in Exp. 1 returns* 1*, is negligible (as a function of* κ*).*

Experiment $\mathsf{Unforgeability}^{\mathsf{SSS}}_{\mathcal{A}}(\kappa)$
$(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^\kappa)$
$(\mathsf{pk}_{san}, \mathsf{sk}_{san}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^\kappa)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdots),\mathsf{Proof}(\cdot,\mathsf{sk}_{sig},\cdots),\mathsf{Sanit}(\cdots,\mathsf{sk}_{san})}(\mathsf{pk}_{sig}, \mathsf{pk}_{san})$
for $i = 1, 2, \ldots q$ let $(m_i, ADM_i, \mathsf{pk}_{san,i})$ and σ_i
denote the queries and answers to and from oracle Sign
for $j = q+1, \ldots, r$ let $(m_j, MOD_j, \sigma_j, \mathsf{pk}_{sig,j})$ and (m'_j, σ'_j)
denote the queries and answers to and from oracle Sanit
return 1, if
$\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 1$ and
$\forall i \in \{1, 2, \ldots, q\} : (\mathsf{pk}_{san}, m^*) \neq (\mathsf{pk}_{san,i}, m_i)$ and
$\forall j \in \{1, 2, \ldots, q'\} : (\mathsf{pk}_{sig}, m^*) \neq (\mathsf{pk}_{sig,j}, m'_j)$

Experiment 1: Unforgeability experiment for SSSs based on [BFF+09, GQZ11]

Immutability for SSS

Immutability prevents a sanitizer from modifying any block(s) not marked as admissible for this sanitizer by the signer. That is, a sanitizer must not be able to alter a given message in a malicious way. In particular, it must only be able to alter *admissible* blocks. Hence, also deleting or appending blocks must be prohibited.

The following definition is from [PS15], where $ADM(MOD) = 1$ means that MOD only contains modifications which are admissible.

Definition 79 (SSS Immutability [PS15])**.** *An SSS is immutable, if for any efficient adversary $\mathcal{A}$ the probability that the experiment depicted in Exp. 2 returns 1, is negligible (as a function of κ).*

Experiment $\mathsf{Immutability}^{\mathsf{SSS}}_{\mathcal{A}}(\kappa)$
$(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^\kappa)$
$(m^*, \sigma^*, \mathsf{pk}^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{Proof}(\mathsf{sk}_{sig},\cdot,\cdot,\cdot,\cdot)}(\mathsf{pk}_{sig})$
for $i = 1, 2, \ldots, q$ let $(m_i, \mathsf{pk}_{san,i}, ADM_i)$ index the queries to oracle Sign
return 1, if
$\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*) = 1$, and
$\forall i \in \{1, 2, \ldots, q\} : \mathsf{pk}^* \neq \mathsf{pk}_{san,i}$ or $m^* \notin \{MOD(m_i) \mid MOD \text{ with } ADM_i(MOD) = 1\}$

Experiment 2: Immutability experiment for SSSs [PS15]

Privacy for SSS

This definition follows from the original one given in [BFF+09], and the idea is based on the indistinguishability notion for encryption, i.e., the adversary shall not be able recover sanitized

parts of the message. We note that it only considers outsiders as privacy adversaries, i.e., the adversary has no knowledge of any private key(s). See Sect. 5.3.2 for a stronger notion. Note, in standard privacy the adversary is not able to generate any signatures itself nor does it not know the secret key sk_{sig}.

Definition 80 (Privacy for SSS [BFF+09])**.** *A* SSS *is* private*, if for any efficient (PPT) adversary $\mathcal{A}$ the probability that the experiment $\mathsf{Privacy}_{\mathcal{A}}^{SSS}(\kappa)$ given in Exp. 3 returns* 1 *is negligibly close to $\frac{1}{2}$ (as a function of κ).*

Experiment $\mathsf{Privacy}_{\mathcal{A}}^{\mathsf{SSS}}(\kappa)$
$(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^{\kappa})$
$(\mathsf{pk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^{\kappa})$
$b \leftarrow \{0,1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}_{sig},\cdots),\mathsf{Sanit}(\cdots,\mathsf{sk}_{san})}_{\mathsf{Proof}(\mathsf{sk}_{sig},\cdots),\mathsf{LoRSanit}(\cdots,\mathsf{sk}_{sig},\mathsf{sk}_{san},b)}(\mathsf{pk}_{sig}, \mathsf{pk}_{san})$
where oracle LoRSanit on input of
$m_{0,i}, MOD_{0,i}, m_{1,i}, MOD_{1,i}, ADM_i$:
if $MOD_{0,i} \not\subseteq ADM_i$, return $\perp$
if $MOD_{1,i} \not\subseteq ADM_i$, return $\perp$
if $MOD_{0,i}(m_{0,i}) \neq MOD_{1,i}(m_{1,i})$, return $\perp$
let $(m_i, \sigma_i) \leftarrow \mathsf{Sign}(m_{b,i}, \mathsf{sk}_{sig}, \mathsf{pk}_{san}, ADM_i)$
return $(m_i', \sigma_i') \leftarrow \mathsf{Sanit}(m_i, MOD_{b,i}, \sigma, \mathsf{pk}_{sig}, \mathsf{sk}_{san})$
return 1, if $a = b$

Experiment 3: Privacy experiment for SSS based on [BFF+09]

Note, the adversary is able to define the input to the LoRSanit oracle. However, the inner working, in particular the signature σ_i, is not given to the adversary [BFF+09].

Strong Privacy for SSS

This property has been introduced in [dMPPS14]. The experiment in Exp. 4 for strong privacy only differs from the standard privacy experiment (see Sect. 5.3.2 and Exp. 3) in one facet: The secret sanitizer key sk_{san} is given to the adversary $\mathcal{A}$ as a third parameter.

Definition 81 (Strong Privacy for SSS [dMPPS14])**.** *A* SSS *is* strongly private*, if for any efficient adversary $\mathcal{A}$ the probability that the experiment $\mathsf{StrongPrivacy}_{\mathcal{A}}^{SSS}(\kappa)$ given in Exp. 4 returns* 1 *is negligibly close to $\frac{1}{2}$ (as a function of κ).*

In the strong privacy game (cf. Exp. 4) the adversary can use signature and proof oracles to generate and verify signed messages, as in the standard privacy definition. The difference is that the adversary does no longer need the oracle for sanitizing, as it can sanitize messages using sk_{san}. To breach privacy, the adversary has to decide which of two input messages was randomly chosen and used by the oracle to produce the signed outcome. The adversary gives two crafted input messages and modification instructions to the LoRSanit oracle. Note that—as in the standard privacy definition—the adversary is still not generating any of the challenge sanitizations itself. Again, trivial ways to win the game are excluded.

Experiment $\mathsf{StrongPrivacy}_{\mathcal{A}}^{\mathsf{SSS}}(\kappa)$
$(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) \leftarrow \mathsf{KeyGen}sig(1^{\kappa})$
$(\mathsf{pk}_{san}, \mathsf{sk}_{san}) \leftarrow \mathsf{KeyGen}san(1^{\kappa})$
$b \leftarrow \{0,1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\mathsf{pk}_{san},\cdot),\mathsf{Proof}(\mathsf{sk}_{sig},\cdots,\mathsf{pk}_{san}),\mathsf{LoRSanit}(\cdots,\mathsf{sk}_{sig},\mathsf{sk}_{san},b)}(\mathsf{pk}_{sig}, \mathsf{pk}_{san}, \mathsf{sk}_{san})$
where oracle $\mathsf{LoRSanit}$ on input of:
$m_0, MOD_0, m_1, MOD_1, ADM$
if $MOD_0(m_0) \neq MOD_1(m_1)$, return $\perp$
if $MOD_0 \not\subseteq ADM$, return $\perp$
if $MOD_1 \not\subseteq ADM$, return $\perp$
let $(m, \sigma) \leftarrow \mathsf{Sign}(m_b, \mathsf{sk}_{sig}, \mathsf{pk}_{san}, ADM)$
return $(m', \sigma') \leftarrow \mathsf{Sanit}(m, MOD_b, \sigma, \mathsf{pk}_{sig}, \mathsf{sk}_{san})$
return 1, if $a = b$

Experiment 4: Strong privacy experiment for SSS [dMPPS14]

Unlinkability for SSS

Unlinkability makes it infeasible for third parties to link sanitized message-signature pairs to their source message-signature pairs. In the following we present the definition of unlinkability:

Definition 82 (Unlinkability for SSS [BFLS10])**.** *A* SSS *is* unlinkable *if for any efficient algorithm $\mathcal{A}$ the probability that the following experiment $\mathsf{Unlinkability}_{\mathcal{A}}^{\mathsf{SSS}}(\kappa)$ in Exp. 5 returns 1 is negligibly close to* 1/2 *(as a function of κ).*

Strong Unlinkability for SSS

In [BPS13], a strengthened unlinkability notion for sanitizable signatures was presented. The difference of this notion to the conventional unlinkability notion is as follows: The $\mathsf{LoRSanit}$ oracle does not have a fixed signer key pk_{sig} and the adversary is allowed to choose pk_{sig}. This essentially means that unlinkability even holds for signers. We refer the reader to [BPS13] for a discussion of the exact benefits, e.g., being able to guarantee unlinkability and therefore privacy is preserved even if the signer looses its secret key.

Definition 83 (Strong Unlinkability for SSS [BPS13])**.** *A* SSS *is* strongly unlinkable*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{StrongUnlinkability}_{\mathcal{A}}^{\mathsf{SSS}}(\kappa)$ given in Exp. 6 returns* 1 *is negligibly close to* $\frac{1}{2}$ *(as a function of κ).*

Remark 1. *Information leakage through the message itself is out of scope.*

Remark 2. *Unlinkability is stronger than privacy.*

Experiment $\mathsf{Unlinkability}^{\mathsf{SSS}}_{\mathcal{A}}(\kappa)$
$(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^\kappa)$
$(\mathsf{pk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^\kappa)$
$b \leftarrow \{0,1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}_{sig},\cdots),\mathsf{Sanit}(\cdots,\mathsf{sk}_{san})}_{\mathsf{Proof}(\mathsf{sk}_{sig},\cdots),\mathsf{LoRSanit}(\cdots,\mathsf{sk}_{san},\mathsf{sk}_{sig},b)}(\mathsf{pk}_{sig}, \mathsf{pk}_{san})$
where oracle $\mathsf{LoRSanit}$ on input of:
$m_{0,i}, MOD_{0,i}, \sigma_{0,i}, m_{1,i}, MOD_{1,i}, \sigma_{1,i}$
$//ADM$ needs to be recoverable from all σ
if $ADM_{0,i} \neq ADM_{1,i}$, return $\perp$
if $MOD_{0,i} \not\subseteq ADM_{0,i}$, return $\perp$
if $MOD_{1,i} \not\subseteq ADM_{1,i}$, return $\perp$
if $MOD_{0,i}(m_{0,i}) \neq MOD_{1,i}(m_{1,i})$, return $\perp$
if $\mathsf{Verify}(m_{0,i}, \sigma_{0,i}, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) \neq 1$ or
$\mathsf{Verify}(m_{1,i}, \sigma_{1,i}, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) \neq 1$, return $\perp$
return $(m', \sigma') \leftarrow \mathsf{Sanit}(m_{b,i}, MOD_{b,i}, \sigma_{b,i}, \mathsf{pk}_{sig}, \mathsf{sk}_{san})$
return 1, if $a = b$

Experiment 5: Unlinkability experiment for SSSs based on [BFLS10]

Experiment $\mathsf{StrongUnlinkability}^{\mathsf{SSS}}_{\mathcal{A}}(\kappa)$
$(\mathsf{pk}_{san}, \mathsf{sk}_{san}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^\kappa)$
$b \leftarrow \{0,1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sanit}(\cdots,\mathsf{sk}_{san}),\mathsf{LoRSanit}(\cdots,\mathsf{sk}_{san},b)}(\mathsf{pk}_{san})$
where oracle $\mathsf{LoRSanit}$ on input of
$m_0, MOD_0, \sigma_0, m_1, MOD_{1,i}, \sigma_1, \mathsf{pk}_{sig}$:
if $ADM_0 \neq ADM_1$, return $\perp$
if $MOD_0 \not\subseteq ADM_0$, or $MOD_1 \not\subseteq ADM_1$, or $MOD_0(m_0) \neq MOD_1(m_1)$, return $\perp$
if $\mathsf{Verify}(m_0, \sigma_0, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) \neq 1$ or $\mathsf{Verify}(m_1, \sigma_1, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) \neq 1$, return $\perp$
return $(m', \sigma') \leftarrow \mathsf{Sanit}(m_b, MOD_b, \sigma_b, \mathsf{pk}_{sig}, \mathsf{sk}_{san})$
return 1, if $a = b$

Experiment 6: Strong unlinkability experiment for SSS from [BPS13]

Transparency for SSS

Transparency prevents third parties, e.g., verifiers, to decide whether the signer or the sanitizer created a certain (valid) message-signature pair (m, σ).

In a transparent scheme, the adversary as a verifier is not able to decide whether the message-signature pair carries a signature created through Sanit or a freshly produced signature generated by Sign. The property of transparency is a stronger privacy notion, but not required in all applications of SSS (cf. [BPS12]). However, it is of importance if the existence of a sanitizer has to be hidden. For example if visible sanitizations imply disadvantages for any involved party

as it has been discussed in [SPPdM12a, dMPPS13, BFF+09].

A transparent SSS still offers the possibility to determine the accountability for a message by the execution of the Judge algorithm. The algorithm Judge requires additional input from the Proof algorithm. The adversary can not run the Proof algorithm itself as it requires the signer who has access to the needed secret. To keep the adversary as strong as possible, the adversary is given access to signing and sanitizing oracles, as well as a proof oracle. Hence, the adversary can control the input and observe the output of all secret based algorithms.

Definition 84 (Transparency for SSS [BFF+09])**.** *A* SSS *is* proof-restricted transparent*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Transparency}_{\mathcal{A}}^{SSS}(\kappa)$ given in Exp. 7 returns* 1 *is negligibly close to* $\frac{1}{2}$ *(as a function of* κ*).*

Remark 3. *For obvious reasons, the access to the proof-oracle is limited, i.e., its use is excluded for the given* (m, σ)*.*

Remark 4. *Information leakage through the message itself is out of scope.*

Remark 5. *Transparency is stronger than privacy.*

Experiment $\mathsf{Transparency}_{\mathcal{A}}^{\mathsf{SSS}}(\kappa)$
$(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^{\kappa})$
$(\mathsf{pk}_{san}, \mathsf{sk}_{san}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^{\kappa})$
$b \leftarrow \{0, 1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdots),\mathsf{Sanit}(\cdots,\mathsf{sk}_{san}),\mathsf{Proof}(\mathsf{sk}_{sig},\cdots),\mathsf{Sanit/Sign}(\cdots,\mathsf{sk}_{sig},\mathsf{sk}_{san},b)}(\mathsf{pk}_{sig}, \mathsf{pk}_{san})$
where Sanit/Sign for input m, MOD, ADM:
$\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}_{sig}, \mathsf{pk}_{san}, ADM)$,
$(m', \sigma') \leftarrow \mathsf{Sanit}(m, MOD, \sigma, \mathsf{pk}_{sig}, \mathsf{sk}_{san})$
if $b = 1$:
$\sigma' \leftarrow \mathsf{Sign}(m', \mathsf{sk}_{sig}, \mathsf{pk}_{san}, ADM)$,
finally return (m', σ').
if $\mathcal{A}$ has queried any message output by Sanit/Sign to Proof,
return a random bit
else,
return 1, if $a = b$

Experiment 7: Transparency experiment for SSS from [BFF+09]

Accountability: Interactive Non-Public Signer Accountability for SSS

Signer accountability prohibits a sanitizer from being able to blame the signer if indeed the sanitizer is responsible for a given message.

Definition 85 (Interactive Non-Public Signer-Accountability for SSS ([BFF+09, GQZ11]))**.** *A* SSS *is* interactive non-public signer accountable*, if for any efficient algorithm $\mathcal{A}$ the probability*

that the experiment Sig-Acc$^{SSS}_{\mathcal{A}}(\kappa)$ *given in Exp. 8 returns* 1 *is negligible (as a function of* κ*) and additionally the* Judge *algorithm requires a proof* π *generated by* Proof *needing* sk_{sig} *as input to the* Proof*.*

Experiment Sig-Acc$^{\mathsf{SSS}}_{\mathcal{A}}(\kappa)$
$(\mathsf{pk}_{san}, \mathsf{sk}_{san}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^{\kappa})$
$b \leftarrow \{0,1\}$
$(\mathsf{pk}^*, \pi^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sanit}(\cdots, \mathsf{sk}_{san})}(\mathsf{pk}_{san})$
for $i = 1, \ldots, q$ let (m'_i, σ'_i)
denote the answers and queries from and to the oracle Sanit
return 1, if:
$\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}^*, \mathsf{pk}_{san}) = 1$, and
$\forall i \in \{1, \ldots, q\} : (\mathsf{pk}^*, m^*, ADM^*) \neq (\mathsf{pk}_{sig,i}, m'_i, ADM_i)$, and
$\mathsf{Judge}(m^*, \sigma^*, \mathsf{pk}^*, \mathsf{pk}_{san}, \pi^*) = \mathtt{San}$

Experiment 8: Interactive non-public signer accountability experiment for SSS based on [BFF+09, GQZ11]

The interactive non-public signer accountability experiment in Fig. 8 has been extended according to [GQZ11], i.e., the admissible parts ADM are explicitly stated inside the game.

Accountability: Interactive Non-Public Sanitizer Accountability for SSS

sanitizer accountability prohibits a signer from being able to blame the sanitizer if indeed the signer is responsible for a given message.

Definition 86 (Interactive Non-Public Sanitizer-Accountability for SSS ([BFF+09, GQZ11]))**.** *A sanitizable signature scheme* SSS *is* interactive non-public sanitizer accountable*, if for any efficient algorithm* $\mathcal{A}$ *the probability that the experiment* San-Acc$^{SSS}_{\mathcal{A}}(\kappa)$ *given in Exp. 9 returns* 1 *is negligible (as a function of* κ*) and additionally the* Judge *algorithm requires a proof* π *generated by* Proof *algorithm needing* sk_{sig} *as input to the* Proof*.*

Interactive non-public sanitizer accountability are extended according to [GQZ11] by explicitly stating the description of admissible parts ADM inside the games.

Public Non-Interactive Accountability for SSS

A sanitizable signature scheme satisfies *non-interactive public accountability*, if and only if for a valid message-signature pair (m, σ), a third party can correctly decide whether (m, σ) originates from the signer or from the sanitizer without interacting with the signer or sanitizer, i.e., just from using public knowledge $(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san})$.

This notion has been introduced in [BPS12] to obtain the same verification workflow that is present in regular digital signatures schemes and in particular to argue about the legal value of

Experiment $\mathsf{San\text{-}Acc}_{\mathcal{A}}^{\mathsf{SSS}}(\kappa)$
$(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^{\kappa})$
$b \leftarrow \{0,1\}$
$(\mathsf{pk}^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdots,\mathsf{sk}_{sig},\cdots),\mathsf{Proof}(\mathsf{sk}_{sig},\cdots)}(\mathsf{pk}_{sig})$
let $(m_i, ADM_i, \mathsf{pk}_{san,i})$ and σ_i for $i = 1, \ldots, q$
denote the queries to the oracle Sign
$\pi \leftarrow \mathsf{Proof}(\mathsf{sk}_{sig}, m^*, \sigma^*, \{(m_i, \sigma_i) \mid 0 < i \leq q\}, \mathsf{pk}^*)$
return 1, if:
$\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*) = 1$, and
$\forall i \in \{1, \ldots, q\} : (\mathsf{pk}^*_{san}, m^*, ADM^*) \neq (\mathsf{pk}_{san,i}, m_i, ADM_i)$, and
$\mathsf{Judge}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*, \pi) = \mathtt{Sig}$

Experiment 9: Interactive non-public sanitizer accountability experiment for SSS based on [BFF+09, GQZ11]

evidence created by SSS. In public non-interactive accountable schemes, the algorithm Judge must decide correctly even on an empty proof π. As only Proof needs a secret key, this captures the required form of public accountability.

Definition 87 (Non-Interactive Public Accountability for SSS [BPS12])**.** *A sanitizable signature scheme SSS is* non-interactive publicly accountable*, if* $\mathsf{Proof} = \perp$ *and if for any efficient algorithm* $\mathcal{A}$ *the probability that the experiment* $\mathsf{Pubaccountability}_{\mathcal{A}}^{SSS}(\kappa)$ *given in Exp. 10 returns* 1 *is negligible (as a function of* κ*). The basic idea is that an adversary, i.e., the sanitizer or the signer, has to be able to make the Judge decide wrongly on an empty proof* π*.*

Like with the interactive accountability notions, an adversary, e.g., the sanitizer or the signer, has to be able to make the algorithm Judge decide incorrectly in their favour. This results in the requirement that the adversary must either provide a message-signature pair (m^*, σ^*) for which Judge decides `San` even if it was never been output by Sanit or decide `Sig` even if it was never been output by Sign.

The authors also introduced this accountability notion for the level of blocks in [BPS12]. We skip this, as we will subsequently discuss the work in [dMPPS13], which introduces the generalisation of this notion: groups of blocks.

Group-level Signer/Sanitizer Accountability for SSS

In [dMPPS13], the authors define accountability with a detail of group-level, while fully preserving transparency on the message level. The notion is given in [dMPPS13] and continues the initial work in [BPS12]. Informally, a SSS offers *group-level accountability*, if for all valid message-signature pairs (m, σ) the algorithm Proof outputs a proof π which allows the algorithm GJud to decide, if the given *group*-signature pair $(\mathrm{GRP}[i], \sigma)$ originates from the signer or from the sanitizer (even in the presence of malicious signers or sanitizers).

The authors of [BPS12] derived the notion of *block-level* non-interactive public accountability,

Experiment $\mathsf{Pubaccountability}^{\mathsf{SSS}}_{\mathcal{A}}(\kappa)$
$(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^\kappa)$
$(\mathsf{pk}_{san}, \mathsf{sk}_{san}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^\kappa)$
$(\mathsf{pk}^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_{san})}(\mathsf{pk}_{sig}, \mathsf{pk}_{san})$
Let $(m_i, ADM_i, \mathsf{pk}_{san,i})$ and σ_i for $i = 1, 2, \ldots, k$
be the queries and answers to and from oracle Sign
return 1 if
$(\mathsf{pk}^*, m^*) \neq (\mathsf{pk}_{san,i}, m_i)$ for all $i = 1, 2, \ldots, k$, and
$\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*) = 1$, and
$\mathsf{Judge}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*, \perp) = \mathtt{Sig}$
Let $(m_j, MOD_j, \sigma_j, \mathsf{pk}_{sig,j})$ and (m'_j, σ'_j) for $j = 1, 2, \ldots, k'$
be the queries and answers to and from oracle Sanit
return 1 if
$(\mathsf{pk}^*, m^*) \neq (\mathsf{pk}_{sig,j}, m'_j)$ for all $j = 1, 2, \ldots, k'$, and
$\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}^*, \mathsf{pk}_{san}) = 1$, and
$\mathsf{Judge}(m^*, \sigma^*, \mathsf{pk}^*, \mathsf{pk}_{san}, \perp) = \mathtt{San}$
return 0.

Experiment 10: Non-interactive public accountability for SSS from [BPS12]

i.e., a third party can decide which party is accountable for *each block* $m[i]$. The following definitions already target groups of blocks.

Definition 88 (Group-Level Non-Interactive Public Accountability for SSS ([dMPPS13])). *A sanitizable signature scheme SSS (together with an algorithm Detect) is* group-level non-interactive publicly accountable*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment* $\mathsf{Group-Pub-Acc}^{SSS}_{\mathcal{A}}(\kappa)$ *given in Exp. 11 returns* 1 *is negligible (as a function of* κ*).*

The additional Detect algorithm allows to non-interactively and publicly decide who is accountable for a given block and is defined in [dMPPS13] as follows with $d \in \{\mathtt{Sig}, \mathtt{San}, \perp\}$:

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

Additionally, [dMPPS13] introduced a fine-grained scope for interactive non-public accountability, which allows to achieve transparency as well as block-level accountability. This is an improvement over the paper from [BPS12], who initially introduced the paradigm of treating properties on the block-level. In [BPS12], they required to sacrifice transparency.

The additional algorithm GJud is defined in [dMPPS13] as follows:

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

This algorithm GJud adapts the standard accountability property achieved via the Judge algorithm on the message-level to a scope of individual blocks. Properties for each block individually

can be emulated by the above if each block is in its own new group. Message-level properties, e.g., like in [BFF+09], are emulated by having just one single group containing all admissible blocks.

Experiment $\mathsf{Group}-\mathsf{Pub}-\mathsf{Acc}_{\mathcal{A}}^{\mathsf{SSS}}(\kappa)$
$(\mathsf{pk}_{sig},\mathsf{sk}_{sig})\leftarrow\mathsf{KGen}_{\mathrm{sig}}(1^{\kappa})$
$(\mathsf{pk}_{san},\mathsf{sk}_{san})\leftarrow\mathsf{KGen}_{\mathrm{san}}(1^{\kappa})$
$(\mathsf{pk}^*,m^*,\sigma^*)\leftarrow\mathcal{A}^{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdot,\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_{san})}(\mathsf{pk}_{san},\mathsf{pk}_{sig})$
Let $(m_i,ADM_i,\mathsf{pk}_{san,i},\mathrm{GRP}_i)$ and (m_i,σ_i) for $i=1,2,\ldots,k$
be the queries and answers to and from oracle Sign.
Let $(m_j,MOD_j,\sigma_j,\mathsf{pk}_{sig,j})$ and (m'_j,σ'_j) for $j=1,2,\ldots,k'$
be the queries and answers to and from oracle Sanit.
return 1 if
$\mathsf{Verify}(m^*,\sigma^*,\mathsf{pk}_{sig},\mathsf{pk}^*)=1$, and
$\exists q:\mathsf{Detect}(m^*,\sigma^*,\mathsf{pk}_{sig},\mathsf{pk}^*,q)=\mathtt{Sig}$
$(\mathrm{GRP}[q]^*,\mathsf{pk}^*)$ was never queried to Sign
as a group of any m_i queried
return 1, if
$\mathsf{Verify}(m^*,\sigma^*,\mathsf{pk}^*,\mathsf{pk}_{san})=1$, and
$\exists q:\mathsf{Detect}(m^*,\sigma^*,\mathsf{pk}^*,\mathsf{pk}_{san},q)=\mathtt{San}$
$(\mathrm{GRP}[q]^*,\mathsf{pk}^*)$ was never queried to Sanit
as a group of any MOD_i
return 0

Experiment 11: Group-level non-interactive public accountability experiment for SSS from [dMPPS13]

Definition 89 (Group-level Signer Accountability for SSS [dMPPS13])**.** *A sanitizable signature scheme SSS is group-level signer accountable, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment* $\mathsf{Group}-\mathsf{Signer}-\mathsf{Acc}_{\mathcal{A}}^{SSS}(\kappa)$ *given in Exp. 12 returns* 1 *is negligible (as a function of κ). Basically, to win the game the adversary has to generate a tuple* $(\mathsf{pk}^*,m^*,\sigma^*,\pi^*)$*, which leads* GJudge *to decide that the sanitizer is accountable for a group* $\mathrm{GRP}[q]\in m^*$*, while it is not.*

Definition 90 (Group-level Sanitizer Accountability for SSS [dMPPS13])**.** *A sanitizable signature scheme SSS is group-level sanitizer accountable, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment* $\mathsf{Group}-\mathsf{Sanitizer}-\mathsf{Acc}_{\mathcal{A}}^{SSS}(\kappa)$ *given in Exp. 13 returns* 1 *is negligible (as a function of κ). Basically, to win the game the adversary has to generate a tuple* $(\mathsf{pk}^*,m^*,\sigma^*)$ *for which* Proof *generates a proof π which leads* Judge *to decide that the signer is accountable for a group* $\mathrm{GRP}[q]\in m^*$*, while it is not.*

5.3.3 Extensions

For sanitizable signatures we finally want to briefly discuss some extensions. Firstly, extensions that limit the power of the sanitizer in the sense that the signer can define various limitations.

Experiment $\mathsf{Group-Signer-Acc}_{\mathcal{A}}^{\mathsf{SSS}}(\kappa)$
$(\mathsf{pk}_{san}, \mathsf{sk}_{san}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^{\kappa})$
$b \leftarrow \{0,1\}$
$(\mathsf{pk}^*, \pi^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_{san})}(\mathsf{pk}_{san})$
Let $(m_j, MOD_j, \sigma_j, \mathsf{pk}_{sig,j})$ and (m'_j, σ'_j) for $j = 1, 2, \ldots, k$
be the queries and answers to and from oracle Sanit.
return 1, if:
$\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}^*, \mathsf{pk}_{san}) = 1$, and
$\exists q : \mathsf{GJud}(m^*, \sigma^*, \mathsf{pk}^*, \mathsf{pk}_{san}, \pi^*, q) = \mathtt{San}$
$(\mathrm{GRP}[q]^*, \mathsf{pk}^*)$ was never queried to Sanit
as a group of any MOD_i

Experiment 12: Group-level signer accountability experiment for SSS from [dMPPS13]

Experiment $\mathsf{Group-Sanitizer-Acc}_{\mathcal{A}}^{\mathsf{SSS}}(\kappa)$
$(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^{\kappa})$
$b \leftarrow \{0,1\}$
$(\mathsf{pk}^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{Proof}(\mathsf{sk}_{sig},\cdot,\cdot,\cdot,\cdot)}(\mathsf{pk}_{sig})$
Let $(m_i, MOD_i, \sigma_i, \mathsf{pk}_{san,i}, \mathrm{GRP}_i)$ and (m_i, σ_i) for $i = 1, 2, \ldots, k$
be the queries and answers to and from the oracle Sign.
$\pi \leftarrow \mathsf{Proof}(\mathsf{sk}_{sig}, m^*, \sigma^*, \{(m_i, \sigma_i) | 0 < i \leq q\}, \mathsf{pk}^*)$
return 1, if:
$\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}^*, \mathsf{pk}_{san}) = 1$, and
$\exists q : \mathsf{GJud}(m^*, \sigma^*, \mathsf{pk}^*, \mathsf{pk}_{san}, \pi^*, q) = \mathtt{Sig}$ and
$(\mathrm{GRP}[q]^*, \mathsf{pk}^*)$ was never queried to Sign
as a group of any m_i

Experiment 13: Group-level sanitizer accountability experiment for SSS from [dMPPS13]

Secondly, extensions to the model of sanitizable signatures that support multiple entities (signers and sanitizers) and a seperation of signing and declaring the sanitizer(s).

Extended Sanitizable Signatures (ESSS). In [KL06], Klonowski and Lauks informally proposed several extensions to sanitizable signatures that allow to reduce the sanitizer's power, which were later formalized in [CJ10].

They proposed extensions (1) to limit the possible modifications to sets of allowed modifications per message block (`LimitSet`), (2) to force the sanitizer to make the same changes in logically linked blocks (`EnforceModif`), (3) to limit the number of modified blocks (`LimitNbModif`) upon a single sanitization and (4) to limit the number of possible versions of a sanitized message (`LimitNbSanit`).

Unfortunately, the formalization in [CJ10] does not define privacy in the original sense of sanitizable signatures, i.e., schemes which reveal the sets of allowed modifications for the `LimitSet`

blocks upon verification of a (sanitized) signature can be proven to be private in their model. To fix this issue, [DS15] strengthened the privacy notion for ESSS to reflect privacy in the original sense. Besides that, [DS15] showed that ESSS fulfilling this stronger privacy notion can be black-box constructed from any secure SSS in the model of [BFF+09] and indistinguishable accumulators [DHS15].[19]

Sanitizable Signatures for Multiple Signers and Sanitizers. In [CJL12], the model of sanitizable signatures has been extended from the single-user setting to the multi-user setting. In particular, this generalized model supports multiple signers and sanitizers.

Trapdoor Sanitizable Signatures. In [CLM08] the authors introduced so called trapdoor sanitizable signatures, where—instead of delegating to one or many sanitizers at the time of signing—the signer can delegate the sanitization rights to (potentially) multiple sanitizers by issuing a trapdoor at any time after the signature has been produced. This is achieved by introducing an additional Trapdoor algorithm that—given a message, a signature and a secret key—produces a trapdoor. An improved instantiation has later been proposed in [YSL10]. Since the constructions of [CLM08, YSL10] loose the accountability property, later in [LDW13] accountable trapdoor sanitizable signatures have been introduced.

5.4 Redactable Signatures

In the case of redactable signature schemes (RSS) the modification of a signed message is limited to the removal or blackening of parts (blocks) of a document without invalidating the signature. Redactable signature schemes have been independently introduced in [JMSW02, SBZ01], and are the basis for many subsequent works.

Here it is not required that the deletion is undetectable. One might think of a special symbol, e.g., "■", as a substitution of the previous information to denote that a redaction has taken place. If the redaction is detectable, then there can be different levels of detection, e.g., the length of the redacted content, or the position of the redacted blocks relative to the rest of the message may be visible.

5.4.1 Formal Definition

The set of algorithms present in this section have been formalised in [BBD+10] initially for tree-structured documents.

In the following, the entity of the signer always denotes the original signer of a given message m and the redactor denotes an entity that is authorised to redact at least one block of this m (typically any party is allowed to redact). Note, this section fixes the message's structure to ordered lists of message blocks for RSS*s* without loss of generality, i.e., an RSS in general can be

[19] We note that [DS15] is currently an unpublished manuscript, which will be made available on request.

defined to work on any form of message. Regardless of its structure or even on fully or partially unstructured message it is assumed that the actual RSS defines how the message is decomposed into blocks.

Definition 91 (Redactable Signature Scheme)**.** *An* RSS *consists of at least four PPT algorithms:* (*KeyGen*, *Sign*, *Redact*, *Verify*)*. Note, all algorithms may output* $\perp$ *in case of an error.*

Key Generation.

There is a key generation algorithm for the signer, which creates one pair of keys consisting of a private key and the corresponding public key, where κ *denotes the security parameter:*

$$(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) \leftarrow \mathit{KeyGen}(1^{\kappa}).$$

Signing.

The Sign *algorithm takes as input a message* $m = (m[1], \ldots, m[\ell])$*, with each block* $m[i] \in \{0,1\}^*$*, the signer's secret key* sk_{sig}*. It outputs a signature* σ *and the message (or* $\perp$ *on error):*

$$(m, \sigma) \leftarrow \mathit{Sign}(m, \mathsf{sk}_{sig}).$$

It is assumed that ADM is always recoverable from any signature $\sigma \neq \perp$ *by the sanitizer.*

Redact.

Algorithm Redact *takes a message* $m = (m[1], \ldots, m[\ell])$*, with each block* $m[i] \in \{0,1\}^*$*, a modification instruction MOD, a signature* σ*, the signer's public key* pk_{sig}*. It modifies the message* m *according to the modification instruction MOD, which contains a set of block indices* $(j_0, \ldots, j_k)$ *for those blocks that shall be removed. The algorithm* Redact *generates an updated signature* σ' *for the modified message* $m' \leftarrow MOD(m)$*. Then* Redact *outputs* m' *and* σ' *(or* $\perp$ *in case of an error).*

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

Note, the algorithm Redact *requires no secret keys.*

Verification.

The Verify *algorithm outputs a decision as a bit* $d \in \{1,0\}$ *indicating the validity of the signature* σ *for the message* m *with respect to the signer's public key* pk_{sig}*.*

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

5.4.2 Security Properties

In this section we give an overview of the most common security properties for RSS. For redactable signatures the security properties are correctness, unforgeability, privacy, transparency, accountability, where the latter notion has only recently been introduced for RSS in [PS15].

Correctness of RSS

In a nutshell, the correctness properties for RSS require that every genuinely signed message must verify (signing correctness) and that every valid message that is genuinely redacted again must verify (redaction correctness). Thereby, the *signing correctness* as presented for SSS in [BFF+09] and in [BPS12] can be equivalently applied for RSS. For readability these are restated here in the harmonised notation.

Definition 92 (Signing correctness for RSS (same as for SSS given in [BFF+09])). *All genuinely signed messages (m, σ) are accepted as valid by Verify.*

> *For any security parameter κ, any key pair $(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) \leftarrow \mathit{KeyGen}(1^\kappa)$, any message $m \in \{0,1\}^*$ and any $\sigma \leftarrow \mathit{Sign}(m, \mathsf{sk}_{sig})$ we have*
>
> $$\mathit{Verify}(m, \sigma, \mathsf{pk}_{sig}) = 1.$$

The *sanitizing correctness* defined for SSS in Definition 74 can not be re-used, as RSS misses the notion of the sanitizer's key. We take the *redaction correctness*, that was formalised for tree structured data in [BBD+10].

Definition 93 (Redaction correctness for RSS ([BBD+10])). *All genuinely redacted messages (m, σ) are accepted as valid by Verify.*

> *For any security parameter κ, any key pair $(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathit{KeyGen}(1^\kappa)$, any message $m \in \{0,1\}^*$, any σ with $\mathit{Verify}(m, \sigma, \mathsf{pk}_{sig}) = 1$, any MOD, and any pair $(m', \sigma') \leftarrow \mathit{Redact}(m, MOD, \sigma, \mathsf{pk}_{sig})$, we require*
>
> $$\mathit{Verify}(m', \sigma', \mathsf{pk}_{sig}) = 1.$$

Unforgeability of RSS

Unforgeability guarantees that a valid signature on a message can only be generated by the signer as it requires knowledge of sk_{sig}, or by a redactor, by correctly redacting a message which already carries a valid signature and is in the redactor's possession.

Remark 6. *Valid redactions are excluded from being forgeries.*

Definition 94 (Unforgeability for RSS [BBD+10]). *A RSS is* unforgeable, *if for any efficient (PPT) adversary $\mathcal{A}$ the probability that the game depicted in Exp. 14 returns 1, is negligible (as a function of κ).*

Experiment $\mathsf{Unforgeability}_{\mathcal{A}}^{\mathsf{RSS}}(\kappa)$
$(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) \leftarrow \mathsf{KeyGen}(1^{\kappa})$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, \mathsf{sk}_{sig})}(\mathsf{pk}_{)}$
for $i = 1, \ldots, q$ let m_i denote the queries to oracle Sign
return 1, if
$\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}_{sig}) = 1$ and
$m^* \notin \bigcup_{i=1}^{q} \mathrm{span}_{\vDash}(m_i)$

Experiment 14: Unforgeability experiment for RSS from [BBD+10]

Privacy for RSS

A *private* scheme prevents verifiers from recovering any information (especially the original value) about block(s) of m from redacted block(s) of m', given m' and a valid signature σ' over m'.

Remark 7. *Information leakage through the modified message itself is out of scope.*

Definition 95 (Privacy for RSS [BBD+10])**.** *A redactable signature scheme* RSS *is* private*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Privacy}_{\mathcal{A}}^{\mathsf{RSS}}(\kappa)$ given in Exp. 15 returns* 1 *is negligibly close to* $\frac{1}{2}$ *(as a function of κ).*

Experiment $\mathsf{Privacy}_{\mathcal{A}}^{\mathsf{RSS}}(\kappa)$
$(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) \leftarrow \mathsf{KeyGen}(1^{\kappa})$
$b \xleftarrow{\$} \{0, 1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, \mathsf{sk}_{sig}), \mathsf{LoRRedact}(\cdots, \mathsf{sk}_{sig}, b)}(\mathsf{pk}_{)}$
where oracle $\mathsf{LoRRedact}$
for input m_0, MOD_0, m_1, MOD_1:
if $MOD_0(m_0) \neq MOD_1(m_1)$, return $\perp$
let $(m, \sigma) \leftarrow \mathsf{Sign}(m_b, \mathsf{sk}_{sig})$
return $(m', \sigma') \leftarrow \mathsf{Redact}(m, MOD_b, \sigma, \mathsf{pk}_{sig})$.
return 1, if $a = b$

Experiment 15: Privacy experiment for RSS from [BBD+10]

Transparency for RSS

Transparency prevents third parties, e.g., verifiers, to decide which party (signer or sanitizer) is accountable for a given valid message-signature pair (m, σ).

Remark 8. *Information leakage through the message itself is out of scope.*

Remark 9. *Transparency is stronger than privacy.*

Definition 96 (Transparency for RSS [BBD+10])**.** *An* RSS *is transparent, if for any efficient adversary* $\mathcal{A}$ *the probability that the experiment given in Exp. 16 returns* 1 *is negligibly close to* $\frac{1}{2}$ *(as a function of* κ*).*

In the experiment the adversary chooses the message m to be signed and also the target outcome of the message's redaction by providing MOD. The Redact/Sign oracle then chooses randomly if it outputs the signature generated by signing the supplied message and then redacting it (case $b = 0$) or if it outputs a fresh signature using Sign on the modified message's contents. To win, the adversary has to identify with a non negligible probability if a signature over a supplied message was generated by Redact or Sign.

Experiment $\mathsf{Transparency}^{\mathsf{RSS}}_{\mathcal{A}}(\kappa)$
$(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^{\kappa})$
$b \leftarrow \{0, 1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig}),\mathsf{Redact/Sign}(\cdot,\cdot,\mathsf{sk}_{sig},b)}(\mathsf{pk}_{sig})$
where oracle Redact/Sign for input m, MOD:
$\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}_{sig})$
$(m', \sigma') \leftarrow \mathsf{Redact}(m, MOD, \sigma, \mathsf{pk}_{sig})$
if $b = 1$: $\sigma' \leftarrow \mathsf{Sign}(m', \mathsf{sk}_{sig})$
return (m', σ')
return 1, if $a = b$

Experiment 16: Transparency experiment for RSS based on [BBD+10]

Non-Interactive Public and Interactive Non-Public Accountability for RSS

Contrary to sanitizable signatures (SSS) described previously, accountability of redactions has not been considered in the initial setting. It has only very recently been formalized in [PS15].

Recall, accountability allows to determine who is accountable for a given signature/message pair (σ, m). Accountability makes it possible to deduct, maybe in an interactive protocol with involvement of the signer, if a signature σ was created by the signer or by redaction of a certain dedicated party. This requires an additional key-pair to be given to the redacting party, making this operation no longer public, unless one would make the key public. This still gives accountability in the sense that even in a transparent RSS, the signer could generate a proof to a judge that the message was redacted, if it was indeed redacted.

Accountable redactable signatures (ARSS) as introduced in [PS15] offer "two different characteristics" of accountability: an online form, where the original signer has to generate a proof, and an offline version where anyone is able to determine the accountable party (called public accountability).

The basic idea for a construction of an ARSS according to [PS15] is to use an SSS which makes any RSS accountable by signing the original message m and the corresponding redactable signature σ_{RSS} of that m using the SSS. While m and σ_{RSS} can be changed publicly by redaction

using the RSS, the outer sanitizable signature can only be updated by dedicated and accountable sanitizers using the SSS.

The ARSS needs to have additional algorithms, which are close in behaviour and definition than those from the underlying SSS. For the sake of brevity these are not restated here and we refer the reader to [PS15] for the details.

5.5 Overview of Constructions for Sanitizable and Redactable Signatures

Various constructions of RSS and SSS are provided in literature. Subsequently, we briefly discuss underlying construction principles of some of those schemes and provide an overview in Table 6.

		Sig.	Scheme				
Scheme	R/S	EU	DET	GS	CH	Acc	Agg
[BFLS09]	S	✓†,‡					
[BPS12]	S	✓†,‡					
[BPS13]	S		✓†,‡				
[BFLS10]	S		✓†	✓‡			
[ACdMT05]	S	✓†			✓‡		
[BFF+09]	S	✓†			✓‡		
[dMPPS13]	S	✓†			✓‡		
[CLM08]	S	✓†			✓‡		
[GQZ11]	S	✓†			✓‡		
[BBD+10]	R	✓					
[PSPdM12]	R	✓				✓	
[ABC+12]	R					✓	
[PS14]	R					✓	
[IKO+07]	R						✓
[SPB+12a]	R						✓

Table 6: Overview of sanitizable/redactable signature constructions. Legend: R... redactable signature scheme, S... sanitizable signature scheme, †... S-signer, ‡... S-sanitizer, EU... existentially unforgeable under chosen message attacks (EUF-CMA), DET... deterministic, GS... anonymous non-frameable group signature scheme, CH... chameleon hash, Acc... accumulator, Agg... aggregate signature scheme

Signature Based Schemes. These constructions just require the generation/verification of usually two signatures created by a secure digital signature scheme. The basic idea is that the signer signs the non-admissible, i.e., fixed, block(s) and some meta data identifying, among others, the admissible block(s) or information identifying the dedicated sanitizer. Moreover, the admissible block(s) are then signed either by the signer, in the original or by the sanitizer. While the minimum requirement is that the scheme is EUF-CMA secure, some constructions require additional properties for the underlying signature scheme (cf. Table 6). In particular, the construction from [BPS12] requires just EUF-CMA while [BPS13] additionally requires

the signature to be unique to achieve a stronger notion of privacy, i.e., unlinkability. The construction from [BFLS10] requires a group signature scheme that is anonymous and non-frameable to achieve unlinkability. Other schemes working solely with signatures are [BFLS09, BBD+10]. Besides the aforementioned SSS, one can also trivially construct RSS only from EUF-CMA secure digital signature schemes, with the drawback of a number of signature linear in the number of (admissible) blocks (e.g., the naive constructions in [JMSW02]).

Chameleon-Hash Based Schemes. These constructions also require an EUF-CMA secure signature, but require additionally that admissible block(s) can be modified without invalidating their hash values. This is achieved by means of chameleon-hashes (also known as trapdoor commitments or chameleon blobs [Fis01]). Chameleon-hashes [KR00] are collision-free cryptographic hash-functions in the traditional sense, unless one knows a secret trapdoor. This trapdoor allows to compute arbitrary collisions for that function. Applied to SSS [ACdMT05], the basic construction idea is to replace admissible blocks in the original message with their chameleon-hashes and then to sign this modified message using an EUF-CMA secure signature scheme (using a scheme following the hash-then sign approach 1.3.3). The trapdoor of the chameleon-hash function is then given to the sanitizer. For the used chameleon hash-function it is important to be key-exposure free [AdM04b] and there are various constructions available [ZSnS03, AdM04a, AdM04b, GWX07, CTZ09]. Further schemes that use chameleon-hashing are [CLM08, BFF+09, GQZ11].

Accumulator or Aggregate Signature Based Schemes. Accumulator schemes [Bd93, DHS15] are schemes that allow to accumulate a finite set into a succinct value called the accumulator. Then, for every element in the set, one can efficiently compute a so called witness which certifies membership of the value in the accumulator, i.e, given the element the witness and the accumulator everybody can check the membership of the element (so called universal accumulators [LLX07] additionally support non-membership witnesses). Such accumulator schemes are a versatile tool, and allows using the membership witness function to verify remaining elements after redacting the element and the witness [ABC+12, PSPdM12, PS14]. These constructions either solely use accumulators [ABC+12, PS14] or use an EUF-CMA secure signature scheme to sign the accumulator [PSPdM12]. The scheme in [ABC+12] achieves a very strong privacy notation, i.e., strong context-hiding, by being limited to "quoting of substrings" rather than redaction of any block. Schemes based on aggregate signatures [BGLS03c], basically either require every redactor (the signer is considered the initial redactor) to sign the respective (redacted) document and aggregate it to the previous signature of the document [IKO+07] or sign every single block of a document, aggregate the signatures and remove the signature of redacted blocks from the aggregate [SPB+12a].

Hash-Tree Based Schemes. Initial works on redactable signatures [SBZ01, JMSW02] use a Merkle hash-tree (or generalizations [SR10]) and consider the leaves as commitments (e.g., randomized hash values) to the message blocks where an EUF-CMA secure signature scheme is used to sign the root hash of the Merkle tree. Note that these constructions must not be confused with redactable signature for tree-structured data [BBD+10, SPB+12b].

5.5.1 Other Constructions With Different Security Models

We note that there are several constructions, realizing what we have denoted as sanitizable and redactable signature schemes, but using other terminology and —more importantly— another security model. It has not been the focus of this report to investigate relations between these different concepts. However, for the sake of completeness we mention a line of work in sanitization (which would be redaction in our terminology) and under different security models [MIM+05, IKTY05, MHI08, HHH+08, YSLM08, NTKK09, HHM+11] and another line of work of redactable signatures under different security notions [CDHK15]. Additionally, there are other works considering schemes for document structures of trees and graphs [KB08, KB10, KAB12], which have been shown to not achieve Brzuska et al.'s privacy [BBD+10] or offer too weak integrity protection [SPB+12b, SPB+12a].

5.6 Related Concepts

Finally, for the sake of completeness we want to briefly discuss other classes of homomorphic (algebraic) signatures which are not considered that relevant in context of PRISMACLOUD.

5.6.1 Transitive Signatures

Transitive signatures have been introduced in [MR02]. The basic idea is to have a signature scheme that allows to dynamically build an authenticated graph, edge by edge. Basically, if one has a signature for edges $\{i, j\}$ and $\{j, k\}$ one is able to efficiently derive a signature for edge $\{i, k\}$ without requiring the secret singing key. Let us consider a graph $G = (V, E)$ and let the signer compute signatures for every edge $\{i, j\} \in E \subseteq V \times V$, then the signer has actually produced a signature for the graph being the transitive closure $G^* = (V^*, E^*)$ of graph G, where $V^* = V$ and $\{i, j\}$ is an edge of E^* if there is path from i to j in G. Now, everybody being only in possession of the public key of the signer and the signatures of edges in E can derive a signature for any edge in E^*. For a various different constructions of transitive signatures for undirected graphs we refer the reader to [BN05].

While constructions for undirected graphs are well studied, it is still an open issue to design transitive signatures for directed graphs. So far, in the directed setting, only constructions for directed trees are known [Yi07, Nev08, CH12].

5.6.2 (Graph-Based) Algebraic Signatures for Sets

In [HM02], so called graph-based algebraic signatures have been introduced. They represent algebraic signatures constructed from graph-based one-time signatures. They represent signatures on sets and are algebraic with respect to union and subset operations and also union and super-intersection operations, where the super-intersection of two sets A and B is the collection of all sets S such that such that $A \cap B \subseteq S \subseteq A \cup B$.

We note that [JMSW02] (who also proposed redactable signatures already discussed extensively in this section) also proposed algebraic signatures that are homomorphic with respect to union and subset operations.

6 Conclusion

In this report we have reviewed the state-of-the-art of functional and malleable signature schemes. More precisely, in this report we have presented the following:

We have reviewed all existing general frameworks that cover functional as well as malleable signature schemes.

We have reviewed certain classes of functional signature schemes that provide efficient constructions usable in a practical setting. More precisely, we have presented proxy signature schemes, blank digital signature schemes, policy-based signature schemes as well as attribute-based signature schemes. Due to known relations of certain types of functional signature schemes to other variants of digital signatures, we have also briefly presented related concepts such as group signature schemes and traceable signature schemes.

We have reviewed linearly homomorphic signature schemes, homomorphic signature schemes for polynomial functions, fully homomorphic signature schemes, and homomorphic aggregate signature schemes as the representatives of malleable signature schemes for arithmetics. As first results within the PRISMACLOUD project, besides reviewing the state-of-the-art, we have investigated the use of aforementioned signature schemes in context of electronic voting, smart grids, and electronic health records (cf. [TDB15]). As one of the most important directions for future research with respect to malleable signature schemes for arithmetics, we consider the efficiency of the constructions. This is especially important when it comes to schemes that support large families of functions such as the very recent fully homomorphic signature schemes.

We have reviewed redactable and sanitizable signature schemes as the two important representatives of malleable signature schemes for editing. As first results within the PRISMACLOUD project we have closed the gap that the property of accountability was not considered for RSS yet and have proposed the notion of accountable RSS [PS15]. As open questions remain investigations between the various different models used in the literature and black-box constructions among them.

The results presented within this report show that functional and malleable signatures are a fruitful field of research. In particular it shows that they have attracted increasing interest and that there have been a number of results within the recent years. Overall, the current state of research for various classes of functional and malleable signature schemes already presents a solid foundation for practical use, where for other classes there are still many open issues. Our goal within PRISMACLOUD is to close the gap to the practical application of all these classes of schemes.

List of References

[ABC+12] J. H. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, B. Waters, et al. Computing on authenticated data. In *Theory of Cryptography*, pages 1–20. Springer, 2012.

[ABC+15] J. H. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, A. Shelat, and B. Waters. Computing on authenticated data. *J. Cryptology*, 28(2):351–395, 2015.

[ACdMT05] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik. Sanitizable signatures. In *Proceedings of ESORICS*, pages 159–177, 2005.

[ACHO11] M. Abe, S. S. M. Chow, K. Haralambiev, and M. Ohkubo. Double-Trapdoor Anonymous Tags for Traceable Signatures. In *ACNS*, volume 6715 of *LNCS*, pages 183–200. Springer, 2011.

[ACJT00] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, pages 255–270, 2000.

[AdM04a] G. Ateniese and B. de Medeiros. Identity-based chameleon hash and applications. In *Financial Cryptography*, pages 164–180, 2004.

[AdM04b] G. Ateniese and B. de Medeiros. On the Key Exposure Problem in Chameleon Hashes. In *4th International Conference on Security in Communication Networks*, LNCS 3352, pages 165–179. Springer, 2004.

[AFG+10] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 209–236, 2010.

[AL11] N. Attrapadung and B. Libert. Homomorphic network coding signatures in the standard model. In *Public Key Cryptography–PKC 2011*, pages 17–34. Springer, 2011.

[ALP12] N. Attrapadung, B. Libert, and T. Peters. Computing on authenticated data: New privacy definitions and constructions. In *Advances in Cryptology–ASIACRYPT 2012*, pages 367–385. Springer, 2012.

[ALP13] N. Attrapadung, B. Libert, and T. Peters. Efficient completely context-hiding quotable and linearly homomorphic signatures. In *Public-Key Cryptography–PKC 2013*, pages 386–404. Springer, 2013.

[BB04] D. Boneh and X. Boyen. Short signatures without random oracles. In *Advances in Cryptology-EUROCRYPT 2004*, pages 56–73. Springer, 2004.

[BB08] D. Boneh and X. Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, 2008.

[BB11] D. Boneh and X. Boyen. Efficient selective identity-based encryption without random oracles. *Journal of Cryptology*, 24(4):659–693, 2011.

[BBD+10] 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 *Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010. Proceedings*, pages 87–104, 2010.

[BBG05] D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 440–456, 2005.

[BBS04] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology–CRYPTO 2004*, pages 41–55. Springer, 2004.

[BBW07] J. Bethencourt, D. Boneh, and B. Waters. Cryptographic methods for storing ballots on a voting machine. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2007, San Diego, California, USA, 28th February - 2nd March 2007*, 2007.

[BCC04] E. F. Brickell, J. Camenisch, and L. Chen. Direct Anonymous Attestation. In *ACM CCS*, pages 132–145. ACM, 2004.

[BCCT13] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 111–120, 2013.

[Bd93] J. Benaloh and M. de Mare. One-way Accumulators: A Decentralized Alternative to Digital Signatures. In *EUROCRYPT*, pages 274–285, 1993.

[BDF+14] M. Backes, Ö. Dagdelen, M. Fischlin, S. Gajek, S. Meiser, and D. Schröder. Operational Signature Schemes. *IACR Cryptology ePrint Archive*, 2014:820, 2014.

[BF01] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 213–229, 2001.

[BF11a] D. Boneh and D. M. Freeman. Homomorphic signatures for polynomial functions. In *Advances in Cryptology–EUROCRYPT 2011*, pages 149–168. Springer, 2011.

[BF11b] D. Boneh and D. M. Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In *Public Key Cryptography–PKC 2011*, pages 1–16. Springer, 2011.

[BF14] M. Bellare and G. Fuchsbauer. Policy-Based Signatures. In *PKC 2014*, LNCS. Springer, 2014.

[BFF+09] C. Brzuska, M. Fischlin, T. Freudenreich, A. Lehmann, M. Page, J. Schelbert, D. Schröder, and F. Volk. Security of Sanitizable Signatures Revisited. In *PKC'09*, volume 5443 of *LNCS*, 2009.

[BFG+13] D. Bernhard, G. Fuchsbauer, E. Ghadafi, N. P. Smart, and B. Warinschi. Anonymous Attestation with User-Controlled Linkability. *Int. J. Inf. Sec.*, 12(3):219–249, 2013.

[BFKW09] D. Boneh, D. Freeman, J. Katz, and B. Waters. Signing a linear subspace: Signature schemes for network coding. In *Public Key Cryptography–PKC 2009*, pages 68–87. Springer, 2009.

[BFLS09] C. Brzuska, M. Fischlin, A. Lehmann, and D. Schröder. Santizable Signatures: How to Partially Delegate Control for Authenticated Data. In *BIOSIG 2009*, volume 155 of *LNI*, pages 117–128. Springer, 2009.

[BFLS10] C. Brzuska, M. Fischlin, A. Lehmann, and D. Schröder. Unlinkability of Sanitizable Signatures. In *Public Key Cryptography - PKC 2010*, volume 6056 of *LNCS*, pages 444–461. Springer, 2010.

[BFS14] X. Boyen, X. Fan, and E. Shi. Adaptively secure fully homomorphic signatures based on lattices. 2014.

[BGI+12] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

[BGI14] E. Boyle, S. Goldwasser, and I. Ivan. Functional Signatures and Pseudorandom Functions. In *Public-Key Cryptography - PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, 2014.

[BGLS03a] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology—EUROCRYPT 2003*, pages 416–432. Springer, 2003.

[BGLS03b] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in cryptology—EUROCRYPT 2003*, pages 416–432. Springer, 2003.

[BGLS03c] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *EUROCRYPT*, pages 416–432, 2003.

[BMS13] M. Backes, S. Meiser, and D. Schröder. Delegatable Functional Signatures. Cryptology ePrint Archive, Report 2013/408, 2013. http://eprint.iacr.org/.

[BMW03] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general

assumptions. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, pages 614–629, 2003.

[BN05] M. Bellare and G. Neven. Transitive signatures: new schemes and proofs. *IEEE Transactions on Information Theory*, 51(6):2133–2151, 2005.

[BP97] N. Baric and B. Pfitzmann. Collision-free Accumulators and Fail-stop Signature Schemes Without Trees. In *EUROCRYPT*, pages 480–494, 1997.

[BPS12] C. Brzuska, H. C. Pöhls, and K. Samelin. Non-interactive Public Accountability for Sanitizable Signatures. In *Public Key Infrastructures, EuroPKI 2012*, volume 7868 of *LNCS*, pages 178–193. Springer, 2012.

[BPS13] C. Brzuska, H. C. Pöhls, and K. Samelin. Efficient and Perfectly Unlinkable Sanitizable Signatures without Group Signatures. In *Public Key Infrastructures, EuroPKI 2013*, volume 8341 of *LNCS*, pages 12–30. Springer, 2013.

[BPW12] A. Boldyreva, A. Palacio, and B. Warinschi. Secure Proxy Signature Schemes for Delegation of Signing Rights. *J. Cryptology*, 25(1):57–115, 2012. Extended Version: Cryptology ePrint Archive 2003/096.

[BSZ05] M. Bellare, H. Shi, and C. Zhang. Foundations of Group Signatures: The Case of Dynamic Groups. In *CT-RSA*, volume 3376 of *LNCS*, pages 136–153. Springer, 2005.

[CDHK15] J. Camenisch, M. Dubovitskaya, K. Haralambiev, and M. Kohlweiss. Composable & modular anonymous credentials: Definitions and practical constructions. *IACR Cryptology ePrint Archive*, 2015:580, 2015.

[CFW11] D. Catalano, D. Fiore, and B. Warinschi. Adaptive pseudo-free groups and applications. In *Advances in Cryptology–EUROCRYPT 2011*, pages 207–223. Springer, 2011.

[CFW12] D. Catalano, D. Fiore, and B. Warinschi. Efficient network coding signatures in the standard model. In *Public Key Cryptography–PKC 2012*, pages 680–696. Springer, 2012.

[CFW14] D. Catalano, D. Fiore, and B. Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In *Advances in Cryptology–CRYPTO 2014*, pages 371–389. Springer, 2014.

[CH12] P. Camacho and A. Hevia. Short transitive signatures for directed trees. In *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings*, pages 35–50, 2012.

[Cho09] S. S. M. Chow. Real Traceable Signatures. In *SAC*, volume 5867 of *LNCS*, pages 92–107. Springer, 2009.

[CJ10] S. Canard and A. Jambert. On Extended Sanitizable Signature Schemes. In *CT-RSA'10*, volume 5985 of *LNCS*, 2010.

[CJL09] D. Charles, K. Jain, and K. Lauter. Signatures for network coding. *International Journal of Information and Coding Theory*, 1(1):3–14, 2009.

[CJL12] S. Canard, A. Jambert, and R. Lescuyer. Sanitizable Signatures with Several Signers and Sanitizers. In *Progress in Cryptology - AFRICACRYPT 2012*, volume 7374 of *LNCS*, pages 35–52. Springer, 2012.

[CKLM12] M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn. Malleable proof systems and applications. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 281–300, 2012.

[CKLM14] M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn. Malleable Signatures: New Definitions and Delegatable Anonymous Credentials. In *IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014*, pages 199–213, 2014.

[CL06] M. Chase and A. Lysyanskaya. On signatures of knowledge. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, pages 78–96, 2006.

[CLM08] S. Canard, F. Laguillaumie, and M. Milhau. Trapdoor Sanitizable Signatures and Their Application to Content Protection. In *Applied Cryptography and Network Security, ACNS 2008*, volume 5037 of *LNCS*, pages 258–276. Springer, 2008.

[CLT15] J.-S. Coron, T. Lepoint, and M. Tibouchi. New multilinear maps over the integers. Technical report, Cryptology ePrint Archive, Report 2015/162, 2015. http://eprint.iacr.org, 2015.

[CLY09] J. Cathalo, B. Libert, and M. Yung. Group encryption: Non-interactive realization in the standard model. In *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 179–196, 2009.

[CM11] S. Chatterjee and A. Menezes. On cryptographic protocols employing asymmetric pairings - the role of ψ revisited. *Discrete Applied Mathematics*, 159(13):1311–1322, 2011.

[CMP14] D. Catalano, A. Marcedone, and O. Puglisi. Authenticating computation on groups: New homomorphic primitives and applications. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, pages 193–212, 2014.

[CTZ09] X. Chen, H. Tian, and F. Zhang. Comments and Improvements on Chameleon Hashing Without Key Exposure Based on Factoring. In *IACR Cryptology ePrint Archive*, number 319, 2009.

[CV10] L. Czap and I. Vajda. Signatures for multisource network coding. Technical report, ArXiv, 2010.

[CvH91] D. Chaum and E. van Heyst. Group Signatures. In *EUROCRYPT*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.

[DCNR11] J. Dong, R. Curtmola, and C. Nita-Rotaru. Practical defenses against pollution attacks in wireless network coding. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):7, 2011.

[DFF+13] B. Deiseroth, V. Fehr, M. Fischlin, M. Maasz, N. F. Reimers, and R. Stein. Computing on Authenticated Data for Adjustable Predicates. In *Applied Cryptography and Network Security, ACNS 2013*, volume 7954 of *LNCS*, pages 53–68. Springer, 2013.

[DHS14] D. Derler, C. Hanser, and D. Slamanig. Privacy-Enhancing Proxy Signatures from Non-interactive Anonymous Credentials. In *Data and Applications Security and Privacy XXVIII*, volume 8566 of *LNCS*, pages 49–65. Springer, 2014.

[DHS15] D. Derler, C. Hanser, and D. Slamanig. Revisiting cryptographic accumulators, additional properties and relations to other primitives. In *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, pages 127–144, 2015.

[DKXY03] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong key-insulated signature schemes. In *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, pages 130–144, 2003.

[dMPPS13] H. de Meer, H. C. Pöhls, J. Posegga, and K. Samelin. Scope of Security Properties of Sanitizable Signatures Revisited. In *ARES 2013*, pages 188–197. IEEE, 2013.

[dMPPS14] H. de Meer, H. C. Pöhls, J. Posegga, and K. Samelin. On the Relation between Redactable and Sanitizable Signature Schemes. In *ESSoS 2014*, volume 8364 of *LNCS*, pages 113–130. Springer, 2014.

[DS15] D. Derler and D. Slamanig. Rethinking privacy for extended sanitizable signatures and a black-box construction of strongly private schemes. Unpublished manuscript, 2015.

[EHM11] A. Escala, J. Herranz, and P. Morillo. Revocable attribute-based signatures with adaptive security in the standard model. In *Progress in Cryptology - AFRICACRYPT 2011 - 4th International Conference on Cryptology in Africa, Dakar, Senegal, July 5-7, 2011. Proceedings*, pages 224–241, 2011.

[FHS14] G. Fuchsbauer, C. Hanser, and D. Slamanig. EUF-CMA-Secure Structure-Preserving Signatures on Equivalence Classes. Cryptology ePrint Archive, Report 2014/944, 2014. http://eprint.iacr.org/.

[FHS15] G. Fuchsbauer, C. Hanser, and D. Slamanig. Practical Round-Optimal Blind Signatures in the Standard Model. In *Advances in Cryptology–CRYPTO 2015*. Springer, 2015.

[Fis01] M. Fischlin. *Trapdoor Commitment Schemes and Their Applications*. PhD thesis, Johann Wolfgang Goethe-Universität Frankfurt am Main, 2001.

[FP08] G. Fuchsbauer and D. Pointcheval. Anonymous Proxy Signatures. In *6th International Conference on Security and Cryptography for Networks, SCN'08*, volume 5229 of *LNCS*, pages 201–217. Springer, 2008.

[Fre12] D. M. Freeman. Improved security for linearly homomorphic signatures: A generic framework. In *Public Key Cryptography–PKC 2012*, pages 697–714. Springer, 2012.

[GGH+13] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49, 2013.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GHR99] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology—EUROCRYPT'99*, pages 123–139. Springer, 1999.

[GKKR10] R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin. Secure network coding over the integers. In *Public Key Cryptography–PKC 2010*, pages 142–160. Springer, 2010.

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[GNS10] M. Gagné, S. Narayan, and R. Safavi-Naini. Threshold attribute-based signcryption. In *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*, pages 154–171, 2010.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206. ACM, 2008.

[GQZ11] J. Gong, H. Qian, and Y. Zhou. Fully-Secure and Practical Sanitizable Signatures. In *ISC'11*, volume 6584 of *LNCS*, 2011.

[Gro06] J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, pages 444–459, 2006.

[GS08] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, pages 415–432, 2008.

[GVW14] S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. Cryptology ePrint Archive, Report 2014/897, 2014. http://eprint.iacr.org/.

[GW11] C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 99–108, 2011.

[GWX07] W. Gao, X. Wang, and D. Xie. Chameleon hashes without key exposure based on factoring. *J. Comput. Sci. Technol.*, 22(1):109–113, 2007.

[Her14] J. Herranz. Attribute-based signatures from RSA. *Theor. Comput. Sci.*, 527:73–82, 2014.

[Her15] J. Herranz. Attribute-based versions of schnorr and elgamal. *IACR Cryptology ePrint Archive*, 2015:213, 2015.

[HHH+08] 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 *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008, Tokyo, Japan, March 18-20, 2008*, pages 353–362, 2008.

[HHM+11] G. Hanaoka, S. Hirose, A. Miyaji, K. Miyazaki, B. Santoso, and P. Yang. Sequential bitwise sanitizable signature schemes. *IEICE Transactions*, 94-A(1):392–404, 2011.

[HLC+13] J. Y. Hwang, S. Lee, B.-H. Chung, H. S. Cho, and D. Nyang. Group Signatures with Controllable Linkability for Dynamic Membership. *Inf. Sci.*, 222:761–778, 2013.

[HLhC+11] J. Y. Hwang, S. Lee, B. ho Chung, H. S. Cho, and D. Nyang. Short Group Signatures with Controllable Linkability. In *LightSec*, pages 44–52. IEEE, 2011.

[HLLR12] J. Herranz, F. Laguillaumie, B. Libert, and C. Ràfols. Short attribute-based signatures for threshold predicates. In *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings*, pages 51–67, 2012.

[HM02] A. Hevia and D. Micciancio. The provable security of graph-based one-time signatures and extensions to algebraic signature schemes. In *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, pages 379–396, 2002.

[HMO13] R. Hiromasa, Y. Manabe, and T. Okamoto. Homomorphic signatures for polynomial functions with shorter signatures. In *The 30th Symposium on Cryptography and Information Security, Kyoto*. 2013.

[HS13a] C. Hanser and D. Slamanig. Blank Digital Signatures. In *8th ACM SIGSAC Symposium on Information, Computer and Communications Security (AsiaCCS). Full Version: Cryptology ePrint Archive, Report 2013/130*, pages 95–106. ACM, 2013.

[HS13b] C. Hanser and D. Slamanig. Warrant-Hiding Delegation-by-Certificate Proxy Signature Schemes. In *Progress in Cryptology - INDOCRYPT 2013*, volume 2013 of *LNCS*, pages 60–77. Springer, 2013.

[HS14] C. Hanser and D. Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In *Advances in Cryptology–ASIACRYPT 2014*, pages 491–511. Springer, 2014.

[HW09] S. Hohenberger and B. Waters. Short and stateless signatures from the rsa assumption. In *Advances in Cryptology-CRYPTO 2009*, pages 654–670. Springer, 2009.

[IKO+07] T. Izu, N. Kunihiro, K. Ohta, M. Takenaka, and T. Yoshioka. A sanitizable signature scheme with aggregation. In *ISPEC'07: Proceedings of the 3rd international conference on Information security practice and experience*, pages 51–64, Berlin, Heidelberg, 2007. Springer-Verlag.

[IKTY05] T. Izu, N. Kanaya, M. Takenaka, and T. Yoshioka. PIATS: A partially sanitizable signature scheme. In *Information and Communications Security, 7th International Conference, ICICS 2005, Beijing, China, December 10-13, 2005, Proceedings*, pages 72–83, 2005.

[Jin14] Z. Jing. An efficient homomorphic aggregate signature scheme based on lattice. *MATHEMATICAL PROBLEMS IN ENGINEERING*, 2014.

[JMSW02] R. Johnson, D. Molnar, D. Song, and D. Wagner. Homomorphic signature schemes. In *Topics in Cryptology—CT-RSA 2002*, pages 244–262. Springer, 2002.

[KAB12] A. Kundu, M. J. Atallah, and E. Bertino. Leakage-free redactable signatures. In *Second ACM Conference on Data and Application Security and Privacy, CODASPY 2012, San Antonio, TX, USA, February 7-9, 2012*, pages 307–316, 2012.

[Kat10] J. Katz. *Digital Signatures*. Springer, 2010.

[KB08] A. Kundu and E. Bertino. Structural signatures for tree data structures. *PVLDB*, 1(1):138–150, 2008.

[KB10] A. Kundu and E. Bertino. How to authenticate graphs without leaking. In *EDBT 2010, 13th International Conference on Extending Database Technology, Lausanne, Switzerland, March 22-26, 2010, Proceedings*, pages 609–620, 2010.

[Kha07a] D. Khader. Attribute based group signature with revocation. Cryptology ePrint Archive, Report 2007/241, 2007. http://eprint.iacr.org/.

[Kha07b] D. Khader. Attribute based group signatures. Cryptology ePrint Archive, Report 2007/159, 2007. http://eprint.iacr.org/.

[KL06] M. Klonowski and A. Lauks. Extended Sanitizable Signatures. In *ICISC'06*, volume 4296 of *LNCS*, 2006.

[KMPR05] E. Kiltz, A. Mityagin, S. Panjwani, and B. Raghavan. Append-only signatures. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, pages 434–445, 2005.

[KP06] S. Kunz-Jacques and D. Pointcheval. About the security of MTI/C0 and MQV. In *Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006, Proceedings*, pages 156–172, 2006.

[KR00] H. Krawczyk and T. Rabin. Chameleon Hashing and Signatures. In *Symposium on Network and Distributed Systems Security*, pages 143–154, 2000.

[KTY04] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 571–589, 2004.

[KW93] M. Karchmer and A. Wigderson. On span programs. In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 102–111, 1993.

[KZG10] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 177–194, 2010.

[LAS+10] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren. Attribute-based signature and its applications. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010, Beijing, China, April 13-16, 2010*, pages 60–69, 2010.

[LDW13] J. Lai, X. Ding, and Y. Wu. Accountable trapdoor sanitizable signatures. In *Information Security Practice and Experience - 9th International Conference, ISPEC 2013, Lanzhou, China, May 12-14, 2013. Proceedings*, pages 117–131, 2013.

[LGK+11] S.-H. Lee, M. Gerla, H. Krawczyk, K.-W. Lee, and E. A. Quaglia. Performance evaluation of secure network coding using homomorphic signature. In *Network Coding (NetCod), 2011 International Symposium on*, pages 1–6. IEEE, 2011.

[LJYP15] B. Libert, M. Joye, M. Yung, and T. Peters. Secure efficient history-hiding append-only signatures in the standard model. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pages 450–473, 2015.

[LK10] J. Li and K. Kim. Hidden attribute-based signatures without anonymity revocation. *Inf. Sci.*, 180(9):1681–1689, 2010.

[LLX07] J. Li, N. Li, and R. Xue. Universal Accumulators with Efficient Nonmembership Proofs. In *ACNS*, volume 4521 of *LNCS*, pages 253–269. Springer, 2007.

[LMRS04] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 74–90, 2004.

[LPJY13] B. Libert, T. Peters, M. Joye, and M. Yung. Linearly homomorphic structure-preserving signatures and their applications. In *Advances in Cryptology–CRYPTO 2013*, pages 289–307. Springer, 2013.

[MCVH12] L. Malina, J. Castellà-Roca, A. Vives-Guasch, and J. Hajny. Short-Term Linkable Group Signatures with Categorized Batch Verification. In *FPS*, volume 7743 of *LNCS*, pages 244–260. Springer, 2012.

[Mei09] S. Meiklejohn. An Extension of the Groth-Sahai Proof System. Master's thesis, Brown University, 2009.

[MHI08] K. Miyazaki, G. Hanaoka, and H. Imai. Invisibly sanitizable digital signature scheme. *IEICE Transactions*, 91-A(1):392–402, 2008.

[MIM+05] 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.

[Mol03] D. Molnar. Homomorphic signature schemes. BA Thesis, Harvard College, 2003.

[MOY04] T. Malkin, S. Obana, and M. Yung. The hierarchy of key evolving signatures and a characterization of proxy signatures. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 306–322, 2004.

[MPR08] H. K. Maji, M. Prabhakaran, and M. Rosulek. Attribute-based signatures: Achieving attribute-privacy and collusion-resistance. *IACR Cryptology ePrint Archive*, 2008:328, 2008.

[MPR11] H. K. Maji, M. Prabhakaran, and M. Rosulek. Attribute-based signatures. In *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, pages 376–392, 2011.

[MR02] S. Micali and R. L. Rivest. Transitive signature schemes. In *Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings*, pages 236–243, 2002.

[MUO96] M. Mambo, K. Usuda, and E. Okamoto. Proxy signatures for delegating signing operation. In *ACM Conference on Computer and Communications Security*, pages 48–57. ACM, 1996.

[MVOV96] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[Nev08] G. Neven. A simple transitive signature scheme for directed trees. *Theor. Comput. Sci.*, 396(1-3):277–282, 2008.

[NFW99] T. Nakanishi, T. Fujiwara, and H. Watanabe. A Linkable Group Signature and Its Application to Secret Voting. *Trans. of Information Processing Society of Japan*, 40(7), 1999.

[NTKK09] R. Nojima, J. Tamura, Y. Kadobayashi, and H. Kikuchi. A storage efficient redactable signature in the standard model. In *Information Security, 12th International Conference, ISC 2009, Pisa, Italy, September 7-9, 2009. Proceedings*, pages 326–337, 2009.

[OSEH13] K. Ohara, Y. Sakai, K. Emura, and G. Hanaoka. A Group Signature Scheme with Unbounded Message-Dependent Opening. In *ACM ASIA CCS*, pages 517–522. ACM, 2013.

[OT13] T. Okamoto and K. Takashima. Decentralized attribute-based signatures. In *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, pages 125–142, 2013.

[OT14] T. Okamoto and K. Takashima. Efficient attribute-based signatures for non-monotone predicates in the standard model. *IEEE T. Cloud Computing*, 2(4):409–421, 2014.

[PH09] A. Pfitzmann and M. Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. http://dud.inf.tu-dresden.de/Anon_Terminology.shtml, December 2009. v0.32.

[Pöhed] H. C. Pöhls. *Increasing the Legal Evidentiary Value of Private Malleable Signatures*. PhD thesis, University of Passau, Germany, (to be published).

[PS14] H. C. Pöhls and K. Samelin. On Updatable Redactable Signatures. In *Applied Cryptography and Network Security, ACNS 2014*, volume 8479 of *LNCS*, pages 457–475. Springer, 2014.

[PS15] H. C. Pöhls and K. Samelin. Accountable redactable signatures. In *Proc. of the 10th International Conference on Availability, Reliability and Security (ARES 2015)*. IEEE, Aug. 2015. The original publication is going to be available soon at ieeexplore.ieee.org.

[PSPdM12] H. C. Pöhls, K. Samelin, J. Posegga, and H. de Meer. Flexible redactable signature schemes for trees — extended security model and construction. In *Proc. of the International Conference on Security and Cryptography (SECRYPT 2012)*, pages 113–125. SciTePress, 2012.

[PST13] C. Papamanthou, E. Shi, and R. Tamassia. Signatures of Correct Computation. In *TCC*, volume 7785 of *LNCS*, pages 222–242. Springer, 2013.

[Riv04] R. L. Rivest. On the notion of pseudo-free groups. In *Theory of cryptography*, pages 505–521. Springer, 2004.

[Rom90] J. Rompel. One-Way Functions are Necessary and Sufficient for Secure Signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 387–394, 1990.

[RST01] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, pages 552–565, 2001.

[SBZ01] R. Steinfeld, L. Bull, and Y. Zheng. Content Extraction Signatures. In *ICISC*, volume 2288 of *LNCS*, pages 285–304. Springer, 2001.

[SEH+12] Y. Sakai, K. Emura, G. Hanaoka, Y. Kawai, T. Matsuda, and K. Omote. Group Signatures with Message-Dependent Opening. In *Pairing*, volume 7708 of *LNCS*, pages 270–294. Springer, 2012.

[SMP08] J. C. N. Schuldt, K. Matsuura, and K. G. Paterson. Proxy signatures secure against proxy key exposure. In *Public Key Cryptography - PKC 2008, 11th International Workshop on Practice and Theory in Public-Key Cryptography, Barcelona, Spain, March 9-12, 2008. Proceedings*, pages 141–161, 2008.

[SPB+12a] K. Samelin, H. C. Pöhls, A. Bilzhause, J. Posegga, and H. de Meer. On structural signatures for tree data structures. In *Applied Cryptography and Network Security - 10th International Conference, ACNS 2012, Singapore, June 26-29, 2012. Proceedings*, pages 171–187, 2012.

[SPB+12b] K. Samelin, H. C. Pöhls, A. Bilzhause, J. Posegga, and H. de Meer. Redactable signatures for independent removal of structure and content. In *Information Security Practice and Experience - 8th International Conference, ISPEC 2012, Hangzhou, China, April 9-12, 2012. Proceedings*, pages 17–33, 2012.

[SPPdM12a] K. Samelin, H. C. Pöhls, J. Posegga, and H. de Meer. Block-level Accountability for Transparent Sanitizable Signatures (MIP-1209). Technical report, University of Passau, 12 2012.

[SPPdM12b] K. Samelin, H. C. Pöhls, J. Posegga, and H. de Meer. Redactable vs. Sanitizable Signatures (MIP-1208). Technical report, University of Passau, 12 2012.

[SR10] D. Slamanig and S. Rass. Generalizations and Extensions of Redactable Signatures with Applications to Electronic Healthcare. In *Communications and Multimedia Security, CMS 2010*, volume 6109 of *LNCS*, pages 201–213. Springer, 2010.

[SSU14] D. Slamanig, R. Spreitzer, and T. Unterluggauer. Adding Controllable Linkability to Pairing-Based Group Signatures for Free. In *ISC*, volume 8783 of *LNCS*, pages 388–400. Springer, 2014.

[TDB15] G. Traverso, D. Demirel, and J. Buchmann. Homomorphic signature schemes - a survey. Cryptology ePrint Archive, Report 2015/653, 2015. http://eprint.iacr.org/.

[TV15] S. R. Tate and R. Vishwanathan. Expiration and revocation of keys for attribute-based signatures. Cryptology ePrint Archive, Report 2015/400, 2015. http://eprint.iacr.org/.

[Wan10] Y. Wang. Insecure" provably secure network coding" and homomorphic authentication schemes for network coding. *IACR Cryptology ePrint Archive*, 2010:60, 2010.

[Wat05] B. Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology–EUROCRYPT 2005*, pages 114–127. Springer, 2005.

[WCR11] L. Wei, S. E. Coull, and M. K. Reiter. Bounded vector signatures and their applications. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2011, Hong Kong, China, March 22-24, 2011*, pages 277–285, 2011.

[Wei05] V. K. Wei. Tracing-by-Linking Group Signatures. In *ISC*, volume 3650 of *LNCS*, pages 149–163. Springer, 2005.

[WHW13] F. Wang, Y. Hu, and B. Wang. Lattice-based linearly homomorphic signature scheme over binary field. *Science China Information Sciences*, 56(11):1–9, 2013.

[YCK10] A. Yun, J. H. Cheon, and Y. Kim. On homomorphic signatures for network coding. *IEEE Transactions on Computers*, (9):1295–1296, 2010.

[Yi07] X. Yi. Directed transitive signature scheme. In *Topics in Cryptology - CT-RSA 2007, The Cryptographers' Track at the RSA Conference 2007, San Francisco, CA, USA, February 5-9, 2007, Proceedings*, pages 129–144, 2007.

[YSL10] D. H. Yum, J. W. Seo, and P. J. Lee. Trapdoor Sanitizable Signatures Made Easy. In *Applied Cryptography and Network Security, ACNS 2010*, volume 6123 of *LNCS*, pages 53–68. Springer, 2010.

[YSLM08] T. H. Yuen, W. Susilo, J. K. Liu, and Y. Mu. Sanitizable signatures revisited. In *Cryptology and Network Security, 7th International Conference, CANS 2008, Hong-Kong, China, December 2-4, 2008. Proceedings*, pages 80–97, 2008.

[YWRG08] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan. An efficient signature-based scheme for securing network coding against pollution attacks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008.

[YYLF12] W. Yan, M. Yang, L. Li, and H. Fang. Short signature scheme for multi-source network coding. *Computer Communications*, 35(3):344–351, 2012.

[Zha10] N. Zhang. Signatures for network coding. 2010.

[ZKMH07] F. Zhao, T. Kalker, M. Médard, and K. J. Han. Signatures for content distribution with network coding. In *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pages 556–560. IEEE, 2007.

[ZSnS03] F. Zhang, R. Safavi-naini, and W. Susilo. ID-Based Chameleon Hashes from Bilinear Pairings. In *IACR Cryptology ePrint Archive*, number 208, 2003.

[ZYW12] P. Zhang, J. Yu, and T. Wang. A homomorphic aggregate signature scheme based on lattice. *Chinese Journal of Electronics*, 21(4):701–704, 2012.