# Homomorphic Signature Schemes - A Survey

Giulia Traverso[1], Denise Demirel[2], and Johannes Buchmann[3]

[1] Technische Universität Darmstadt, Germany, gtraverso@cdc.informatik.tu-darmstadt.de
[2] Technische Universität Darmstadt, Germany, ddemirel@cdc.informatik.tu-darmstadt.de
[3] Technische Universität Darmstadt, Germany, buchmann@cdc.informatik.tu-darmstadt.de

**Abstract.** Homomorphic signature schemes are an important primitive for many applications and since their introduction numerous solutions have been presented. Thus, in this work we provide the first exhaustive, complete, and up-to-dated survey about the state of the art of homomorphic signature schemes. First, the general framework where homomorphic signatures are defined is described and it is shown how the currently available types of homomorphic signatures, these are the linearly homomorphic signature schemes, the homomorphic schemes supporting polynomial functions, the fully homomorphic signature schemes, and the homomorphic aggregate signature schemes, can then be derived from such a framework. In addition, this work also presents a description of each of the schemes presented so far together with the properties it provides. Furthermore, three use cases, electronic voting, smart grids, and electronic health records, where homomorphic signature schemes can be employed are described. For each of these applications the requirements that a homomorphic signature scheme should fulfill are defined and the suitable schemes already available are listed. This also highlights the shortcomings of current solutions. Thus, this work concludes with several ideas for future research in the direction of homomorphic signature schemes.

## 1 Introduction

In the last years there has been an increasing interest in homomorphic signature schemes. Thus, many schemes have been proposed that are suitable for a lot of different applications. In this work we overcome the extensive state of the art by presenting a survey of the existing approaches and the properties they provide. In addition, we look into three interesting use cases for homomorphic operations on authenticated data, these are, electronic voting, smart grids, and electronic health records, and identify their requirements. We show to what extend the existing solutions meet these conditions and highlight promising directions for future work.

Homomorphic signature schemes have been initially designed to establish authentication in network coding and to address pollution attacks (see [10]). However, since they allow for computations on authenticated data, they are also a useful primitive for many other applications. In fact, after Johnson et al. introduced a formal definition and a precise framework for homomorphic signatures in 2002 (see [37]), in the following years many schemes have been presented and discussed. The first schemes proposed only allowed to perform linear computations on authenticated data (e.g., [60], [56], [57], and [19]). These approaches have been further improved with respect to efficiency, security, and privacy ([10], [28], [3], [4], [26], [16], [17], [5], [17], [50], and [12]). In addition, to be more flexible, solutions have been developed supporting polynomial functions ([11], [34], [18]), or even coming without any restrictions on the functions themselves, so called fully homomorphic signature schemes ([32], [14]). However, all these solutions assume that each input signature has been signed using the same private key. To overcome this restriction, the homomorphic property has been added to the aggregate signature schemes ([59], [36]) allowing for operations on signatures generated using even different secret-public key pairs.

In this work we start by providing a formal definition of these four types of homomorphic signature schemes. First, the passage from the digital signature schemes to the homomorphic ones is formally described, where the novelties introduced by the homomorphic property itself are highlighted. Afterwards, it is described how to obtain the linearly homomorphic signature schemes from the merely homomorphic ones. And then, starting from the linearly homomorphic signature schemes, it is shown how to derive the ones supporting polynomial functions and how to define the fully ones. Finally, it is formally described how to combine the homomorphic property together with the aggregative one in order to obtain a homomorphic aggregate signature scheme.

Up to our knowledge, this survey is the first such work providing both a description of each single homomorphic signature scheme and a description of the whole general framework in a methodical and didactical approach. Indeed the survey proposed in [58] is not up-to-dated, while in [15] the existing homomorphic signature schemes are just listed, without any deeper discussions. Furthermore, in this survey we also discuss the possible use cases electronic voting, smart grids, and electronic health records. For each use case, concrete examples of how improvements can be achieved by the usage of homomorphic signature schemes are provided, together with the definition of the minimal requirements these schemes should fulfill. Furthermore, it is shown which of the currently existing homomorphic signature schemes are suitable for which of the use cases in question. When that is not the case, directions for future works are proposed.

In Section 2, the definition of general digital signature schemes is recalled and the formal description of the homomorphic signature schemes is provided. In Section 3, the linearly homomorphic signature schemes, the homomorphic signature schemes for polynomial functions, the fully homomorphic signature schemes, and the homomorphic aggregate signature schemes are described. In Section 4, interesting properties for homomorphic signature schemes are discussed. The description of each of the currently existing homomorphic signature scheme and the properties they provide follow in Section 5. In Section 6, the usage of homomorphic signature schemes for each of the aforementioned use cases is presented. Finally, in Section 7 a conclusion is given and possible directions for future work are provided.

## 2 From Digital to Homomorphic Signature Schemes

Generally speaking, a signature is a cryptographic primitive whose aim is to provide:

- *integrity*: protection from non-authorized modifications of the signed message;
- *authenticity*: guarantee of the source, the destination, and the content of the signed message;
- *non-repudiation*: the signer cannot deny to have signed the message.

In this chapter, in order to make clear what a homomorphic signature is, we first provide a definition for digital signature schemes and their security. Afterwards, we present those two definitions for homomorphic signatures.

## 2.1 Digital Signatures

A digital signature scheme is defined over the following sets [31]:

- the messages space $\mathcal{M}$;
- the space of signed messages $\mathcal{Y}$;
- the set of private keys $\mathcal{K}$;
- the set of public keys $\mathcal{K}'$.

In addition, the following three probabilistic, polynomial-time algorithms are defined:

- an algorithm $\mathrm{Set} : 1^\lambda \to \mathcal{K} \times \mathcal{K}'$ for the preliminary stage, where the secret key (used in the signing process) and the public key (used in the verification process) are chosen;
- an algorithm $\mathrm{Sig} : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ for signing;
- a deterministic algorithm $\mathrm{Vrf} : \mathcal{K}' \times \mathcal{M} \times \mathcal{Y} \to \{\mathrm{ok}, \mathrm{bad}\}$ for verification.

For an algorithm, being a *probabilistic, polynomial-time* one means that some randomness is added in the input and the running time is bounded by a polynomial. Also, as a consequence for the randomness in input, the output is probabilistic.

We now specify the role of each of the aforementioned algorithm in the following definition ([45]).

**Definition 1.** *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 $\lambda$ 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 $\sigma$ 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 2.** *A digital signature scheme is* correct *if, for any signature $\sigma$ produced by* Sig *on message $m \in \mathcal{M}$ and private key $k$, then* $\mathrm{Vrf}(k', m, \mathrm{Sig}(k, m)) = \mathrm{ok}$.

## 2.2 Digital Signatures' Security Definition

The security of a digital signature is evaluated taking into account the opponent's attack capacity, who could act in different contexts [31]. If Alice is the signer, Bob the receiver, and Eve the one who wants to forge the signature, from the least dangerous attack to the most sever one, we find:

– *Key-Only Attack.* Eve knows Alice's public key.
– *Message Attacks.* Eve has access to some signature-message pairs. This attack consists of four different options:
  • *Known Message Attack.* Eve knows some pairs $(m_1, \sigma_1), (m_2, \sigma_2), \ldots, (m_k, \sigma_k)$, but $m_1, m_2, \ldots, m_q$ have not been chosen by her.
  • *Generic Chosen Message Attack.* Eve can obtain from Alice some pairs $(m_1, \sigma_1)$, $(m_2, \sigma_2), \ldots, (m_k, \sigma_k)$ for a chosen list of messages $m_1, m_2, \ldots, m_k$. Though this choice is generic, i.e. independent of Alice's public key, the attack is nonadaptive: $m_1, m_2, \ldots, m_q$ are chosen before seeing the signatures.
  • *Directed Chosen Message Attack.* Eve is in the same situation as before, with the only difference that she can chose the messages $m_1, m_2, \ldots, m_q$ after seeing Alice's public key. Therefore it is called directed: the aim is to attack the signer whose public key has been seen.
  • *Adaptive Chosen Message Attack.* Eve can in addition see the signatures before choosing the list of messages $m_1, m_2, \ldots, m_q$: Alice is seen as an oracle, to which Eve has an adaptive access (Eve can now request signatures of messages which depend on already-obtained signatures).

A signature *forgery* is the ability of the attacker to get a pair $(m, \sigma)$, where $\sigma$ has been created by the enemy itself, i.e. the legitimate signer hasn't signed $m$. In the following, we present (again in hierarchical order) the possible results after performing an attack:

– *Total Break.* Eve retrieves the private key.
– *Universal Forgery.* Eve is able to model an algorithm which forges signatures for any message. In short, Eve's model is an algorithm equivalent to Alice's one.
– *Selective Forgery.* The forgery is created on a specific message which has been previously chosen by Eve herself.
– *Existential Forgery.* Eve forges a signature for at least one message; the attacker might get some message-signature pairs $(m, \sigma)$, but she has no control over the messages.

Therefore the strongest notion of security for a digital signature we can achieve is security *against existential forgery under adaptive chosen message attack.*

## 2.3 Homomorphic Signatures

The work by Johnson et al. (see [37]) is the first one which provides a rigorous definition of what a homomorphic signature is. The authors highlight also the main problematics concerning such scheme and define the path of the future research.

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 signed message $\mathcal{Y}$ (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 [37].

**Definition 3.** *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 2.1, for homomorphic signatures there are some new elements to take into account:

- *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 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 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 $\tau$*: 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 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 $\overrightarrow{\sigma}$.

Note that the other three algorithms now behave slightly differently 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 [26].

**Definition 4.** *A* homomorphic signature scheme *is a tuple of the following probabilistic, polynomial-time algorithms:*

- Set $(1^\lambda, N)$. *It takes as input a security parameter $\lambda$ 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 $(k, \tau, m, i)$. *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 secret key $k$, which is the signature for the $i$-th message $m$ of the dataset tagged by $\tau$.*
- Vrf $(k', \tau, m, \sigma, f)$. *It takes as input a public key $k'$, a tag $\tau \in \{0,1\}^\lambda$, a message $m \in \mathcal{M}$, a signature $\sigma \in \mathcal{Y}$, and a function $f \in \mathcal{F}$. It outputs "*ok*" if $\sigma$ is a valid signature for the message $m$, output of the function $f$ over the dataset tagged by $\tau$, under the public key $k'$. It outputs "*bad*" otherwise.*

– Eval $(k', \tau, f, \overrightarrow{\sigma})$. *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 $\overrightarrow{\sigma} \in \mathcal{Y}^N$. It outputs a signature $\sigma' \in \mathcal{Y}$ for a function $f \in \mathcal{F}$ over the (tuple of) signatures $\overrightarrow{\sigma} \in \mathcal{Y}^N$, 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 $\pi_i \in \mathcal{F}, \quad \forall\, i \in \{1, 2, \ldots, N\}$. 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 $\pi_i$.

**Definition 5.** *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 $\sigma$ 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 $(m_i, \sigma_i)$ pairs for which* $\mathrm{Vrf}(k', \tau, m_i, \sigma_i) = \mathrm{ok}$*, with $i \in \{1, 2, \ldots, N\}$,*

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

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

## 2.4 Homomorphic Signatures' Security Definition

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 [37] (the first work on homomorphic signature schemes), 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 $\text{span}_*(m_1, m_2, \ldots, m_N)$. We would like indeed that it is difficult to write a message $m \in \text{span}_*(m_1, m_2, \ldots, m_N)$ 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 [46]), 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 [37]).

We define the *advantage* of an adversary $E$ as the probability that she outputs a valid signature $(m', \sigma')$ for some message $m' \notin \text{span}_*(m_1, m_2, \ldots, m_N)$, after queries on messages $m_1, m_2, \ldots, m_N$.

**Definition 6.** *A homomorphic signature scheme is* $(t, q, \varepsilon)$- secure *against existential forgeries (with respect to* $*$*), if every adversary* $E$ *making no more than* $q$ *chosen-message queries and running in time at most* $t$ *has advantage at most* $\varepsilon$*:*

$$Adv\ E \leq \varepsilon.$$

## 3 Homomorphic Signature Schemes

In this section we present two types of signature schemes satisfying homomorphic properties. In the first subsection we provide a description of those schemes having the homomorphic property only. Afterwards, in the second subsection, we discuss the homomorphic signature schemes presenting also the aggregative property.

### 3.1 Types of Homomorphic Signature Schemes

After providing a formal definition of what a homomorphic signature scheme is, we now discuss three different typologies. In fact, 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 linear homomorphic signature scheme [36] is 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:

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

The definition of linearly homomorphic signatures follows (see [10]).

**Definition 7.** *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 $\lambda$ 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 $\tau$.*
- $\mathrm{Vrf}(k', \tau, v, \sigma, f)$. *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 $\sigma$ is a valid signature for the vector $v$, output of the function $f$ over the dataset tagged by $\tau$, under the public key $k'$. It outputs "*bad*" otherwise.*
- $\mathrm{Eval}(k', \tau, f, \overrightarrow{\sigma})$. *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 $\overrightarrow{\sigma} \in {\mathbb{F}_p}^N$. 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 $\overrightarrow{\sigma} \in {\mathbb{F}_p}^N$, labeled by tag $\tau \in \{0,1\}^\lambda$.*

For the *correctness*, we refer to Definition 5, where the message $m \in \mathcal{M}$ is instead a vector $v \in {\mathbb{F}_p}^N$ and $f(\overrightarrow{m}) = \sum_{i=1}^N c_i v_i$.

**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 [11], 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 [11]. Though, the definition will take into account a maximum degree of the polynomials, which are multivariate.

As described in [11], 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 $f(\overrightarrow{m}) = \sum_{j=1}^{\ell} c_j Y_j(\overrightarrow{m})$ where:

– $\overrightarrow{m} = (m_1, m_2, \ldots, m_N)$;
– $\ell := \binom{N+d}{d} - 1$;
– $\{Y_j\}_{j=1}^{\ell}$ 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 8.** *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 $\lambda$ 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 $\tau$.*
– $\mathrm{Vrf}(k', \tau, m, \sigma, f)$. *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 $\sigma$ is the valid signature for the message $m$, output of the function $f$ over the dataset tagged by $\tau$, under the public key $k'$. It outputs "*bad*" otherwise.*
– $\mathrm{Eval}(k', \tau, f, \overrightarrow{\sigma})$. *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 $\overrightarrow{\sigma} \in R$. It outputs a signature $\sigma' = f(\overrightarrow{\sigma}) \in R$ for a function $f \in \mathcal{F}$ over the (tuple of) signatures $\overrightarrow{\sigma} \in R$, labeled by tag $\tau \in \{0,1\}^{\lambda}$.*

For the *correctness*, we refer to Definition 5, where $f(\overrightarrow{m}) = \sum_{j=1}^{\ell} c_j Y_j(\overrightarrow{m})$.

*Remark 1.* We observe that in [11], the computation of $f \in \mathbb{F}_p[x_1, x_2, \ldots, x_N]$ over the tuple of signatures $\overrightarrow{\sigma}$ 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 $\sigma'$ is the output of $\hat{f}(\overrightarrow{\sigma})$.

**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 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 4). 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. $P(m_1, \dots, m_N) = m_i$. That means that the correctness must hold for single-message signatures (see [14]).

For the *correctness*, due to the generality of the function that a fully homomorphic scheme supports, we refer directly to Definition 5, where the description of such $f$ covers all types of functions.

## 3.2 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 [13]. 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 $u_1, u_2, \dots, u_N$ want to obtain a signature $\sigma$ which is an aggregation of the respective signatures $\sigma_1, \sigma_2, \dots, \sigma_N$. Each signature $\sigma_i$ 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 $\sigma$, it is always possible to aggregate a further one $\sigma_{N+1}$. This means that we don't have to start the process from the scratch and re-set the computations for $\sigma_1, \sigma_2, \dots, \sigma_N, \sigma_{N+1}$. Instead, it is sufficient to generate the final signature $\sigma'$ by aggregating $\sigma_{N+1}$ and $\sigma$.

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 $i \in \{1, 2, \dots, N\}$, 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 $k'_1, k'_2, \dots, k'_N$ (indicated by $\overrightarrow{k'}$), one for each signer;

– a new algorithm is introduced, that is the algorithm $Agg_\sigma$ which aggregates the signatures $\sigma_1, \sigma_2, \dots, \sigma_N$, computing the signature $\sigma$.

A more formal and precise definition follows.

**Definition 9.** *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 $\lambda$ 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 $i \in \{1, 2, \dots, N\}$. 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 $k$.*
– $\mathrm{Vrf}(\overrightarrow{k'}, m, \sigma)$*. It takes as input the public keys' string $\overrightarrow{k'}$, a message $m \in \mathcal{M}$ and a signature $\sigma \in \mathcal{Y}$. It outputs "*ok*" if $\sigma$ is a valid signature for the message $m$, under the public keys $\overrightarrow{k'}$. It outputs "*bad*" otherwise.*
– $\mathrm{Agg}_\sigma(\overrightarrow{k'}, \overrightarrow{m}, \overrightarrow{\sigma})$*. It takes as input a public keys' string $\overrightarrow{k'}$, a messages's string $\overrightarrow{m} \in \mathcal{M}$, and a signatures' string $\overrightarrow{\sigma} \in \mathcal{Y}$. It outputs a signature $\sigma_{agg} \in \mathcal{Y}$, which is the aggregate signature of the signatures in $\overrightarrow{\sigma}$ of the messages in $\overrightarrow{m}$, under the public keys in $\overrightarrow{k'}$, 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 10.** *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 $i \in \{1, 2, \dots, N\}$, if $\sigma$ is the output of* $\mathrm{Sig}(k, m, i)$*, then* $\mathrm{Vrf}(k', m, \sigma) = \mathrm{ok}$.
*(2) If $\sigma_i$ is the output of* $\mathrm{Sig}(k, \tau, m, i)$*, then* $\mathrm{Vrf}(\overrightarrow{k'}, \overrightarrow{m}, \mathrm{Agg}_\sigma(\overrightarrow{k'}, \overrightarrow{m}, \overrightarrow{\sigma})) = \mathrm{ok}$.

*Remark 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 [59], 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 $Agg_\sigma$ 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 11.** *A* homomorphic aggregate signature scheme *is a tuple of the following probabilistic, polynomial-time algorithms:*

- $\mathrm{Set}(1^\lambda, N)$. *It takes as input a security parameter $\lambda$ 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 $f : \mathcal{M}^N \to \mathcal{M}$.*
- $\mathrm{Sig}(k, \tau, m, i)$. *It takes as input a secret key $k$, a tag $\tau \in \{0,1\}^\lambda$, a message $m \in \mathcal{M}$, 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 $m$.*
- $\mathrm{Vrf}(\overrightarrow{k'}, \tau, m, \sigma, f)$. *It takes as input a public keys' string $\overrightarrow{k'}$, 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 $\sigma$ is a valid signature for the message $m$, signed using the public keys $\overrightarrow{k'}$, output of the function $f$ over the dataset tagged by $\tau$. It outputs "*bad*" otherwise.*
- $\mathrm{Agg}_m(\overrightarrow{k'}, \tau, \overrightarrow{m}, f)$. *It takes as input a public keys' string $\overrightarrow{k'}$, a tag $\tau \in \{0,1\}^\lambda$, a messages' string $\overrightarrow{m} \in \mathcal{M}$, 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 $\overrightarrow{m}$ in the dataset labeled by tag $\tau$, coming from the users with public keys $\overrightarrow{k'}$, respectively.*
- $\mathrm{Agg}_\sigma(\overrightarrow{k'}, \tau, \overrightarrow{\sigma}, f)$. *It takes as input a public keys' string $\overrightarrow{k'}$, a tag $\tau \in \{0,1\}^\lambda$, a signatures' string $\overrightarrow{\sigma} \in \mathcal{M}$, and an admissible function $f \in \mathcal{F}$. It outputs a signature $\sigma_{Agg}$, which is the aggregate signature, output of the function $f$ over the signatures $\overrightarrow{\sigma}$ in the dataset labeled by tag $\tau$, coming from the users with public keys $\overrightarrow{k'}$, 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 12.** *A homomorphic aggregate signature is* correct *if for each output of secret-public key pair* $(k, k')$ *of the algorithm* $\mathrm{Set}(1^\lambda, N)$*, the following conditions are valid:*

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

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

**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 $\sigma'$ 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 11, 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 3.* A formal definition of linearly homomorphic aggregate signature is provided in [59] and [36]. Though, let us notice that in literature the linearly homomorphic aggregate signatures are called homomorphic aggregate signatures. 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 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 computation efficiency and shortness of the signature are important features. In other cases these properties might be less important.

In this chapter we discuss and define formally all the above features.

## 4.1 Complexity Assumptions

Basically, so far there are three kinds of hardness assumptions which a homomorphic signature scheme can be based on. They are either built on the Diffie-Hellman problem for 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.

**Bilinear Groups** If $\mathbb{G}$ is an abelian group whose prime order is $p$ and $g \in \mathbb{G}$ is one of its generators, then the *Discrete Logarithm Problem (DLP)* in $\mathbb{G}$ is the following: given $g, g^a \in \mathbb{G}$, find $a \in \mathbb{Z}_p$. However, several cryptosystems are designed over (security equivalent) weaker variants. The most common are the following:

– *Computational Diffie-Hellman Problem (CDH).* On the same assumptions as before, given the triple $(g, g^a, g^b) \in \mathbb{G}$, compute $g^{ab}$.
– *Decisional Diffie-Hellman Problem (DDH).* On the same assumptions as before, given the tuple $(g, g^a, g^b, g^c) \in \mathbb{G}$, decide if in $\mathbb{Z}_p$ it is true that $c = ab$ or not.

Many signature schemes' unforgeabilty proofs rely on hardness assumptions defined over the framework of groups with bilinear maps ([42], [10], [5]). Therefore the above problems have to be adapted to this environment. Let us recall briefly the definition.

**Definition 13.** *A* bilinear group *is a tuple* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, e, \varphi)$ *such that:*

– $\mathbb{G}_1, \mathbb{G}_2$ *and* $\mathbb{G}_t$ *are cyclic groups of prime order* $p$.
– $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ *is bilinear, i.e. for all* $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ *and* $a, b \in \mathbb{Z}, e({g_1}^a, {g_2}^b) = e(g_1, g_2)^{ab}$.
– $e$ *is an* admissible *bilinear map, i.e.*
  • $e$ *is efficiently computable;*
  • *if* $g_1$ *and* $g_2$ *are generators of* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*, respectively, then* $\mathbb{G}_t$ *is generated by* $e(g_1, g_2)$
– *the* Discrete Logarithm Problem *is infeasible to be computed in* $\mathbb{G}_1, \mathbb{G}_2$ *and* $\mathbb{G}_t$.

We list below the several assumptions based on the above definitions.

If in the bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, e, \varphi)$ the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are the same, then we refer to them as $\mathbb{G}$. The bilinear map $e$ becomes $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$. In this situation we can define the Computational Diffie-Hellman Problem.

**Definition 14.** *Given a cyclic group* $\mathbb{G}$ *of prime order* $p$*, the* Computational Diffie-Hellman Problem (CDH) *in* $\mathbb{G}$ *is the following: given* $g, g^a, g^b \in \mathbb{G}$*, compute* $g^{ab} \in \mathbb{G}$.

However, in bilinear groups the decisional Diffie-Hellman assumption is easy to solve (see [39]) and therefore it cannot be used. For such situations, Boneh et al. introduced in [9] the Decisional Linear Problem:

**Definition 15.** *Let* $\mathbb{G}_1$ *be a cyclic group of prime order* $p$ *and a generator* $g_1$ *of* $\mathbb{G}_1$*. Given other arbitrary* $u, v, h$*, all of them generators of* $\mathbb{G}_1$*, the* Decision Linear Problem (DLIN) *in* $\mathbb{G}_1$ *is the following: taken* $a, b, c \in {\mathbb{Z}_p}^*$ *and given as input* $u, v, h, u^a, v^b, h^c \in \mathbb{G}_1$*, output* yes *if* $a + b = c$ *and* no *otherwise.*

The previous problem implies the following Simultaneous Double Pairing Problem ([42]). For simplicity, in the following definition, we denote $\mathbb{G}^3 \backslash \{(1_{\mathbb{G}_t}, 1_{\mathbb{G}_t}, 1_{\mathbb{G}_t})\}$ with ${\mathbb{G}_t}^{*3}$.

**Definition 16.** *Given the pair of cyclic groups $(\mathbb{G}, \mathbb{G}_t)$ of prime order $p$, the* Simultaneous Double Pairing Problem (SDP) *in $(\mathbb{G}, \mathbb{G}_t)$ is the following: given the elements $(g_z, h_z, g_r, h_u) \in \mathbb{G}^4$, it is hard to find a triple $(z, r, u) \in {\mathbb{G}_t}^{*3}$ such that*

$$e(h_z, z) \cdot e(h_u, u) = 1_{\mathbb{G}_t} = e(g_z, z) \cdot e(g_r, r)$$

Another problem is the Flexible Diffie-Hellman Problem, which is slightly stronger than the standard Diffie-Hellman Problem. However it is still a *simple* one, since it implies the fact that distinguishing $g^{abc}$ from a randomly-given tuple $(g, g^a, g^b, g^c)$ is hard [5].

**Definition 17.** *The* Flexible Diffie-Hellman Problem (flexDHP) *for a cyclic group $\mathbb{G}$ with the usual assumptions is the following: given the triple $(g, g^a, g^b)$, where $a, b \in \mathbb{Z}_p$, find another triple $(g^{\mu}, g^{a \cdot \mu}, g^{ab \cdot \mu}) \in \mathbb{G}^3$.*

A variant of the above problem is called $q$-Simultaneous Flexible Pairing Problem, since it is defined in a $q$-fashion.

**Definition 18.** *For a group $\mathbb{G}$ with the usual assumptions, the* $q$-Simultaneous Flexible Pairing ($q$-SFP) *is the following: given a tuple $(g_z, h_z, g_r, h_r, a, \overline{a}, b, \overline{b}) \in \mathbb{G}^8$ and a set of $q$ tuples $(z_j, r_j, s_j, t_j, u_j, v_j, w_j) \in \mathbb{G}^7$ such that*

*(1) $e(a, \overline{a}) = e(g_z, z_j) \cdot e(g_r, r_j) \cdot e(s_j, t_j)$,*
*(2) $e(b, \overline{b}) = e(h_z, z_j) \cdot e(h_r, u_j) \cdot e(v_j, w_j)$,*

*the goal is to find another fresh tuple $(z', r', s', t', u', v', w') \in \mathbb{G}^7$, where $z' \notin \{1_{\mathbb{G}}, z_1, \ldots, z_q\}$, for which (1) and (2) are still valid.*

If the map $e$ runs from distinct groups $\mathbb{G}_1, \mathbb{G}_2$ then we get the co-Computational Diffie-Hellman Problem.

**Definition 19.** *Let $\mathbb{G}_1, \mathbb{G}_2$ and $g_1, g_2$ be as usual. The* co-Computational Diffie-Hellman (co-CDH) *is the following: given $g_1, {g_1}^a \in \mathbb{G}_1$ and $g_2, {g_2}^b \in \mathbb{G}_2$, where $a, b \in \mathbb{Z}_p$, compute ${g_2}^{ab}$.*

Still in the same case, another, more efficient ([5]) assumption, the $q$-Strong Diffie-Hellman Problem, was introduced by Boneh and Boyen [5]. For a security parameter $k \in \mathbb{N}$ and groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order $p > 2^k$, we have that:

**Definition 20.** *Given the triple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t)$, the* q-Strong Diffie-Hellman Problem (q-SDH) *on $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t)$ is the following: for any probabilistic polynomial-time algorithm $E$ and any q= pol(k), the following probability*

$$\Pr[E(g_1, {g_1}^x, {g_1}^{x^2}, \ldots, {g_1}^{x^q}, g_2, {g_2}^x) = (c, {g_1}^{\frac{1}{x+c}})]$$

*is negligible in $k$.*

**Other assumptions**
There are five further complexity assumptions proposed in ([3], [4]). They are defined over the following framework, which is slightly different from the one used so far.

We are again in the case where $\mathbb{G}_1 = \mathbb{G}_2 =: \mathbb{G}$, therefore $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$. The order $p$ of these two groups is not prime, but it is given by $p = p_1 p_2 p_3$, where $p_1, p_2, p_3$ are prime numbers. Let $\mathbb{G}_{p_i}$ be the subgroup of order $p_i$, for $i \in \{1, 2, 3\}$ and let $\mathbb{G}_{p_i p_j}$ be the subgroup of order $p_i p_j$, for $i \neq j$.

*Remark 4.* Let $(u, v) \in \mathbb{G}_{p_i p_j}$ of order $p_i$ and $p_j$, respectively. Then $e(u, v) = 1_{\mathbb{G}_t}$.

- **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 $T \in \mathbb{G}_{p_1 p_2}$ 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 $(g, X_1 X_2, Z_3, Y_2 Y_3)$ and $T$, if $T \in (G)$ or $T \in \mathbb{G}_{p_1 p_3}$.
- **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 $(g, w, g^t, X_1 X_2, X_3, Y_2 Y_3)$ it is hard to decide if $T = w^t Z_3$ or $T = w^t Z_2 Z_3$.
- **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 $(g, g^a, g^b, g^{ab} X_2, X_3, g^c Y_2, Z_2)$ it is hard to compute $e(g, g)^{abc}$.

**RSA** A well known and widely used cryptosystem is RSA. It has been introduced in 1976 in [48] and it is based on the hardness of the *Integer Factorization* problem. Its general underling setting is exploited also for digital and homomorphic signature schemes. In the following, we define the Strong RSA assumption [17].

**Definition 21.** *Given an integer $N = pq$ of length $k$, where $k \in \mathbb{N}$ is the security parameter, $p$ and $q$ are distinct prime numbers, and $z \in \mathbb{Z}_p$, then the* Strong RSA assumption *is the following: for any probabilistic polynomial-time algorithm $E$, the probability*

$$\Pr[(E(N, z) = (y, e) \quad \textit{such that} \quad y^e = z \mod N, e \neq 1)]$$

*is negligible in $k$.*

*Remark 5.* The *RSA assumption* relies on the assumption that the RSA problem is hard. However, the difference between the RSA problem and the stronger RSA assumption is that, in the latter case, the exponent $e$ could be chosen depending on $z$, while in the RSA problem $e$ is chosen independently (see [22]).

**Lattices** Some linear homomorphic signature schemes are built on lattices ([12], [50]). We do not provide a description of lattices, but we introduce the merely assumptions used in the schemes. For an introduction we refer to [12].

**Definition 22.** *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 23.** *Given a matrix* $A \in \mathbb{Z}_q^{n\times m}$ *and* $k$ *short vectors* $x_1, x_2, \dots, x_k \in \mathbb{Z}^m$ *such that* $A \cdot x_i = 0 \mod q$ *for any* $i \in \{1, 2, \dots, k\}$*, the* k-Small Integer Solution (k-SIS) *problem consists of finding another short vector* $x \in \mathbb{Z}^m \backslash \mathbb{Q} - \operatorname{span}\{x_1, x_2, \dots, x_k\}$ *such that* $A \cdot x = 0 \mod q$.

A variant of the above problem, called Inhomogeneous Small Integer Solution, was introduced in [29]. It consists of finding a short solution to a random inhomogeneous system.

**Definition 24.** *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 6.* All the above hardness assumptions can be seen in the $\ell_2$ norm if we assume that there exist a real number $\beta$ such that $||x|| \le \beta$.

## 4.2 Efficiency and Size

Efficiency is a property which still has to be clearly defined in the framework of the homomorphic signature schemes. Indeed, there is no precise standard with respect to comparing all the existing schemes in a rigorous and unique way. Though, it would be desirable as a future work. However, some information and partial comparisons are available in the state of the art, as we will discuss in Chapter 5.

On the other hand, the notion of "*succinctness*" for a signature's size, i.e. the desirable length for a signature, is commonly accepted. For a fixed security parameter $\lambda$, a (homomorphic) signature scheme is called *succinct*, if the signature's length depends only logarithmically on the size $N$ of the dataset (see [11]).

## 4.3 Adversary

Two operative definitions of what security (specifically, unforgeability) is, are provided in the next two subsections. Section 4.3 presents the strong and the more recent one, while Section 4.3 provides a weaker definition that might be sufficient for some applications.

**Boneh et al.'s Security Definition** As we already mentioned, the tag $\tau$ 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 [10]. Since the game presented there is specific for the linear ones, we describe a more general definition given in [11].

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

**Definition 25.** *A homomorphic signature scheme* $\mathcal{S}$ *is* unforgeable *if for all* $N$ *and any probabilistic polynomial-time adversary* $E$*, in the following game the advantage of* $E$ *is negligible with respect to* $\lambda$*:*

– 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* $M$*, the space of signatures* $Y$*, and the set* $F$ *of admissible functions* $f : \mathcal{M}^N \to \mathcal{M}$*. The challenger gives* $k'$ *to* $E$*.*

– Queries*: Proceeding adaptively, the adversary E selects a sequence of data sets $\overrightarrow{m}_i \in \mathcal{M}^N$. 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 $\tau_i$ and the signatures $\sigma_{ij}$, output of* $\mathrm{Sig}(k, \tau_i, m_{ij}, j)$ *for $j \in \{1, 2, \ldots, N\}$, to E.*
– Output*: E outputs:*
  - *a tag $\tau^* \in \{0,1\}^\lambda$;*
  - *a message $m^* \in \mathcal{M}$;*
  - *a signature $\sigma^* \in \mathcal{Y}$.*

The above game formalizes the attacker's intent. Instead of aiming at getting a new pair $(m^*, \sigma^*)$ as for the digital signatures' case, this time E wants to output a triple $(m^*, \sigma^*, f)$ (where $\sigma^*$ is the signature over $f(m^*)$). Being a forgery, the triple is such that it cannot be derived from data and signatures previously seen.

As discussed in [26], the forgery can be done by E 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 [11] for the following formalization.

The adversary E *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 $m^* \neq f(\overrightarrow{m}_i)$.

*Remark 7.* Without the tag $\tau$, 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 [58]).

**Freeman's Security Definition** In [26], Freeman strengthens the adversary: E 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, E might output a triple $(m^*, \sigma^*, f)$ where $\sigma^*$ 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 [26].

## 4.4 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 $\sigma'$ for a message $m'$ is derived. A homomorphic signature scheme is said to be *weakly context hiding* if $\sigma'$ 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 [2], 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 $\sigma'$ has been computed as the output of $\sigma_1, \sigma_2, \ldots, \sigma_N$. This privacy level requires the infeasibility of linking the signature $\sigma'$ to the original ones $\sigma_1, \sigma_2, \ldots, \sigma_q$, even in the case that they are publicly revealed (see [36]).

A further privacy level is introduced in [4]. According to the authors, the definition in [2] takes only the indistinguishability from honestly generated signatures into account. In fact, there are signature schemes (like the one presented in [3]) 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 [36]).

To conclude, there are three notions of privacy for homomorphic signature schemes. They are listed in hierarchical order and the previous one implies the following ones:

– completely context hiding;
– strong context hiding;
– weakly context hiding.

### 4.5 Random Oracle Model vs. Standard Model

In the security proofs (and not only for homomorphic signature schemes) the gap between the ideal framework and the practical one need to be faced. Indeed, in many schemes to be secure, the output of cryptographic primitives involved, usually hash functions, are required to be perfectly random. Though, it is not always possible to specify how such primitives are built. In this case, the security proof is performed in an idealized model called *Random Oracle Model* (see [25]). In the Random Oracle Model, each hash function, for instance, is substituted with a perfectly random function, called random oracle.

Recent schemes are able to perform proofs in a more realistic framework, the *Standard Model*, where perfect randomness is not necessary to prove the security of the schemes themselves. When a signature scheme is set in the Standard Model, that is considered a valuable property.

## 5 State of the Art of Homomorphic Signatures

In this section the state of the art with respect to homomorphic signature schemes is presented. Due to the large number and the different properties they satisfy, they are discussed in separate groups, according to the computations they support. The investigated properties are the ones introduced in the previous sections. Firstly the underlying hardness assumption is specified, then we provide information about the efficiency of the schemes and their signature's length. Afterwards, the general safety of the scheme is discussed: which adversary the signature can cope with and which level of privacy is achieved.

### 5.1 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 [10], it is not the first one in literature. Actually, many schemes had been published before ([19], [60], [56], [57]), but they have already been proven to be not practical or even not completely secure ([51], [58], [24]). 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 [40]). However, in recent works many improvements have been done in this sense.

Since there are several linearly homomorphic signatures, we divide them into two groups. On one hand we have the ones whose security proof relies on the employment of random oracles and on the other hand we have the ones which are proven in the Standard Model, as defined in Section 4.5.

**Random Oracle Model**

**Signing a Linear Subspace: Signature Schemes for Network Coding, by Boneh et al. (2009)** Boneh et al.'s work [10] 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 in $(\mathbb{G}_1, \mathbb{G}_2)$ is infeasible (see Section 4.1). 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 [58]). Furthermore, a lower bound on signatures' length is provided, which shows that the construction is optimal in the sense of [10]. However, being defined over bilinear groups, expensive bilinear maps are needed in order to verify each signature, as shown in [24]. Besides being the first practical scheme, it can cope only with a weak adversary (see Section 4.3). Though, the signature enjoys actually the excellent and valuable property to be completely context hiding in terms of privacy (Section 4.4), as it has been later proven (see [33]).

**Secure Network Coding Over the Integers, by Gennaro et al. (2010)** In 2010, Gennaro et al. presented in [28] a Standard-RSA-based signature scheme (for a definition of the RSA assumption see Section 4.1). Because of the underlying hardness assumption, the signature is less computational expensive than the one presented in [10]. In fact it turned out to be the one providing the lowest computation complexity among the existing schemes in that year. It has also been implemented in Linux, showing that it is actually a practical scheme (see [40]), i.e. the algorithm runs in a reasonable time. The reduction of the bandwidth overhead and the general computational efficiency is possible because the signature works over the integers. Indeed, in order to have a concrete example, the authors of [40] performed an exponentiation by a 1024-bit long exponent running the 512-bit RSA signature scheme proposed in [28] in 3.2 ms. In order to have a fair comparison to the scheme proposed in [10] in terms of the security level, they consider a 112-bit elliptic curve and run the same exponentiation. This computation took 7.79 ms, showing that the scheme proposed by Gennaro et al. is much more efficient and suitable for concrete applications. In fact in the scheme proposed in [28], the linear combinations involve only 8-bit long coefficients, which are much smaller than the 160-bit long coefficients used in [10]. The security is proven under the same definition given in [10]: 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.

**Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures, by Boneh and Freeman (2011)** The signature scheme described in [12] 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 4.1). Efficiency is not clearly discussed. However, being defined over lattices, the scheme is not expected to be as efficient as [28]. This signature scheme is built on the one presented in [30], where the signatures are short vectors in lattices. Therefore also for the signatures provided by [12], 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 8).

**Lattice-Based Linearly Homomorphic Signature Scheme over Binary Fields, by Wang et al. (2013)** The work presented in [50] by Wang et al. improves the first homomorphic schemes defined over binary fields, since it is expected to be resilient against quantum computers [12]. Indeed, the hardness assumption is SIS, as defined in Section 4.1. Furthermore, the scheme is very efficient (both in terms of signing and verifying cost) and provides a short signature and public key. The lattices' parameters are also short, thanks to the usage of the hash function defined in [43], from which the linearity comes from.

As pointed out by the authors of [50], the scheme is secure against Type I forgery and Type II forgery (Section 4.3); thus it is unforgeable only against the weak attacker. Furthermore, the scheme is weakly context hiding, as defined in Section 4.4. With respect to [12], we can then conclude that no improvements have been done in terms of security and privacy.

*Remark 8.* As discussed in [36] 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 [3] 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 4.1) and one computational (Ass. 3, see Section 4.1) are adopted to prove unforgeability. In this case, instead of working with prime fields (like for the scheme described in [10]), the coordinates have to be chosen in $\mathbb{Z}_N$, where $N$ is the composite order of the bilinear groups. Unfortunately, the scheme is inefficient compared to the constructions proven in the Random Oracle Model. The signature is constant in terms of size: it consists of three group elements of at least 1024-bit length. Furthermore, like in [10], the unforgeability of the scheme can only face the weak adversary defined in Section 4.3. Therefore protection against re-randomizing signatures by adversaries is not guaranteed and special countermeasures have to be taken. Note that, according to [26], 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 4.4).

**Computing on Authenticated Data: New Privacy Definitions and Constructions, by Attrapadung et al. (2012)** The same authors of [3] proposed one year later a similar scheme in [4]. The groups $(\mathbb{G}, \mathbb{G}_T)$ considered are of composite order $p = p_1p_2p_3$ and the hardness assumptions are Ass. 1, Ass. 2 , Ass. 4 and Ass. 5, as defined in Section 4.1. This scheme is a direct improvement of the previous one described in [3]: 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 [16] employs and extends the notion of pseudo-free groups introduced by Rivest in [47]. The signature relies on the Strong-RSA assumption (Section 4.1) and actually the authors proved that all the signatures built on RSA are instances of their general framework. Though, the scheme in [16] cannot achieve the same efficiency as the respective scheme in the Random Oracle Model described in [28]. Furthermore, the signature's length is affected by a large random exponent $s$ (for example, for 80-bit of security, $s$ is 1346-bit long) and the signature consists of two integers of at least 1024-bit length.

As for [3], 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. The privacy notion has still to be clarified.

The signature is an element of $\mathbb{Z}_N^*$, where $N$ is the bit size of the RSA modulus $N$. 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 [17], was presented by the same authors of [16] one year later. This new one works over $\mathbb{Z}_N$, where $N = pq$ and $p, q$ are two safe primes. It relies on the same hardness assumption (Strong-RSA) of the scheme described in [16]. 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 [16]. Furthermore, even though it is proven in the Standard Model, the homomorphic signature achieves efficiency comparable to the ones in the Random Oracle Model ([28], [10]). In addition, the number of admissible linear combinations is not bounded anymore like in [16]. 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 [16], in [17] 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 4.1). The scheme can achieve short signatures as well: it consists of one group element and one more element belonging to $\mathbb{Z}_p$ (that is about 512-bit long for 128 bits of security).

In brief, the linearly homomorphic signatures in [17] outperforms in terms of efficiency and size. On the other side, the security hasn't been improved and it is unforgeable under the same definitions of [10] and [16]. In addition, privacy has not been clarified yet.

**Improved Security for Linearly Homomorphic Signatures: A Generic Framework, by Freeman (2012)** In [26], 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 [52] under the CDH, the scheme proposed in [7] under the $q$-Strong DH, the scheme proposed in [27] under the Strong RSA assumption and the scheme proposed in [35] under the RSA assumption. In terms of efficiency and signatures' size, the schemes in [17] achieve better performance. On the other hand, the four schemes in [26] 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 9.* Another important feature of the scheme in [17] is that the to-be-signed vectors's length is unbounded. This is not true for the two constructions of [26] relying on RSA.

**Efficient Completely Context-Hiding Quotable and Linearly Homomorphic Signatures, by Attrapadung et al. (2013)** The work presented in [5] by Attrapadung et al. is a linearly homomorphic signature scheme relying on the Flex-DH hardness assumption (defined in Section 4.1). Such hardness assumption was necessary to improve the signature described in [4] (which is only weakly context hiding) in order to make it completely context hiding. In this way the signature is composed only of groups elements. However, the scheme can cope with the weak adversary only.

In Table 1 we summarize the properties of the linearly homomorphic signatures discussed so far.

**Table 1.** Linearly Homomorphic Signature Schemes

| Signatures: | **Hard. Ass** | **Efficiency** | **Size** | **Adversary** | **Privacy** | **Model** |
|---|---|---|---|---|---|---|
| [**10**] | DHP | acceptable | constant in $N$ | weak | complete | ROM |
| [**28**] | RSA | good | $\emptyset$ | weak | $\emptyset$ | ROM |
| [**12**] | Lattice | $\emptyset$ | short | weak | weak | ROM |
| [**50**] | Lattice | good | constant in $N$ | weak | weak | ROM |
| [**3**] | DHP | bad | 3 groups elem. ($\geq 1024$) | weak | strong | Standard |
| [**4**] | DHP | $\emptyset$ | $33\% \leq$ [3] | *weak* | strong | Standard |
| [**16**] | RSA | $\leq$ [28] | 2 groups elem. ($\geq 1024$) | weak | $\emptyset$ | Standard |
| [**17**] | RSA | good | $\leq$ [16] | weak | $\emptyset$ | Standard |
| [**26**] | DHP/RSA | $\leq$ [17] | $\geq$ [17] | strong | weak | Standard |
| [**5**] | DHP | $\emptyset$ | group elements $\in \mathbb{G}^{16}$ | *weak* | complete | Standard |

## 5.2 Homomorphic Signature Schemes for Polynomial Functions

**Homomorphic Signatures for Polynomial Functions, by Boneh and Freeman (2011)** This signature scheme proposed in [11] is the first homomorphic signature supporting evaluation of multivariate, bounded-degree polynomials on authenticated data. It relies on the SIS hardness assumption and is defined over ideal lattices. Furthermore, it fulfills the definition of succintness of a signature's length, as defined in Section 4.2. In fact the signature depends logarithmically on the data size. On the other hand, the scheme is not very efficient and 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 [34] a signature scheme which improves the one proposed in [11] 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 [18] outperformes both the above signatures ([11], [34]).

As regards the hardness assumptions, the signature in [18] relies on the $k$-Augmented Power Multilinear Diffie-Hellman Problem ($k$-APMDHP). That is a hardness assumption that the authors defined by themselves and we refer directly to [18] for its definition. Despite the fact that lattice-based signatures are in general efficient in terms of computational costs, as remarked in [36], this scheme shows better performance than in [11], at least in terms of verification. However, the scheme relies on multilinear maps and it is still a work-in-progress to define a practical and efficient one (see [21]). 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 [11]. In fact, the signature scheme described in [18] provides two substantial improvements:

(1) the scheme doesn't rely on random oracles any more and the security proof is set 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.

Moreover, the open problem stated by the authors in [11] 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.

In Table 2 we briefly show the signatures' properties discussed so far about the scheme supporting polynomial functions.

**Table 2.** Homomorphic Signature Schemes for Polynomial Functions

| Signatures: | **Hard. Ass** | **Efficiency** | **Size** | **Adversary** | **Privacy** | **Model** |
|---|---|---|---|---|---|---|
| [**11**] | Lattice | bad | $\mathcal{O}\ (d \log N)$ | weak | $\emptyset$ | ROM |
| [**34**] | Lattice | bad | shorter | weak | $\emptyset$ | *ROM* |
| [**18**] | DHP | good | $\mathcal{O}\ (d^3 + d^2 \log N)$ | strong | $\emptyset$ | Standard |

## 5.3 Fully Homomorphic Signature Schemes

**Leveled Fully Homomorphic Signatures from Standard Lattices, by Gorbunov et al. (2014)** In [32], 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 of depth $d$. This scheme bases on the hardness assumption SIS and can evaluate arbitrary circuits over signed data. With respect to efficiency, the costs for verification is as high as computing the function $f$. 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 [8] 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 [14]): 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)** This is the second paper in literature proposing a fully homomorphic signature scheme. It is still based on lattices, assumed to provide security even in the presence of quantum computer. This paper can be thought as a concurrent work to [32]. Indeed some improvements have been done, though arising some problematics 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. On the other hand, the efficiency is claimed to be definitely improved even though there is no discussions about the signature's size. 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.

In short, we summarize the above schemes in Table 3.

**Table 3.** Fully Homomorphic Signature Schemes

| Signatures: | **Hard. Ass** | **Efficiency** | **Size** | **Adversary** | **Privacy** | **Model** | **Note** |
|---|---|---|---|---|---|---|---|
| [**32**] | Lattice | depends on $f$ | poly in depth | weak | weak | Standard | large param. |
| [**32**] | Lattice | depends on $f$ | poly in depth | weak | weak | ROM | short param. |
| [**14**] | Lattice | good | not specified | strong | none | Standard | poly-log depth |

## 5.4 Homomorphic Aggregate Signatures

In many practical applications it might be necessary to aggregate multiple signatures on messages, produced even by different users. The merely linearly homomorphic signatures do not fulfill this new issue: they do not face the multiple users case.

Among the digital signature schemes, the aggregate ones (see [13]) can fulfill this task. 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 new one. Each pair of message-signature comes with an index $i \in 1..N$ referred to the user $i$. That new 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 therefore a quite valuable improvement. Up to our knowledge, there are only two signatures which are both homomorphic and aggregative:

- "*A Homomorphic Aggregate Signature Scheme Based on Lattice*", by Zhang, Yu and Wang (2012) [59]

– “*An Efficient Homomorphic Aggregate Signature Scheme Based on Lattice*”, by Jing (2014) [36]

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, [59] relies on the ISIS problem, while [36] on the SIS one (see Section 4.1). Regarding efficiency, the signature in [36] is better: the signing and the verification costs are improved, leading to a faster scheme. Efficiency is not the only point where the earlier proposed signature in [36] overcomes the older one described in [59]: indeed, while in [59] the signature’s length is *as long as* that of each original signature, in [36] 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 [36] provides also weakly context hiding privacy. Note that this scheme is a variant of the linearly homomorphic signature introduced in [12], where the same privacy property holds and also apply to the multi-users case.

Both schemes are proven in the Random Oracle Model. Thus, a perfectly random, collision-resistant hash function is needed to provide security.

In short, because of the improvements in terms of signature’s size, efficiency and privacy, the signature by [36] is more desirable than the one presented in [59].

Having the aggregative 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 ([23], [54]). In both proposals, the length of the signatures is really short: 128 bits and 160 bits respectively. Though, both of them present the same drawback: they are not efficient. This is also due to the underling hardness problem. While [54] relies on CDH, the first one relies on co-CDH. The several multiplications and exponentiations involved during signing and verification due to pairings and point multiplication over elliptic curves make them not very appealing from practical implementation’s point of view. Thus, we will not further consider these approaches.

The properties of the two existing linearly homomorphic aggregate signatures discussed are reported in Table 4.

**Table 4.** Linearly Homomorphic Aggregate Signature Schemes

| Signatures: | **Hard. Ass** | **Efficiency** | **Size** | **Adversary** | **Privacy** | **Model** |
|---|---|---|---|---|---|---|
| [**59**] | Lattice | good | constant in $N$ | *strong* | $\emptyset$ | *ROM* |
| [**36**] | Lattice | good | constant in $N$ | *strong* | weak | *ROM* |

## 6 Suitable Homomorphic Signature Schemes for Cloud Computing

The signature schemes presented in Section 5 were discussed from an abstract and very general point of view. Now we are going to rephrase what we have said so far highlighting the requirements a scheme needs to fulfill to fit a certain application. Specifically, in this section the review is adapted to the cloud computing setting. In particular, we want to take into account three use cases enabled by the cloud technology. These are: electronic voting, smart grids, and electronic health records. Each of the following sessions is dedicated to one of them. After a brief description of the use case in question, the minimum requirements for a signature are discussed, the available signature schemes are presented, and possible future work is highlighted.

## 6.1 Electronic Voting

Since the existence of democracy, several voting schemes have been designed to let people express their opinion. In order to be *general*, *direct*, *free*, *equal*, and *secret* an election needs to fulfill several security requirements. These include, among others, correctness. More precisely, correctness requires that only votes cast by eligible voters are tallied and that the election outcome is computed correctly without removing and/or adding ballots.

Paper based voting schemes are currently the most widely used ones. Though, they have the drawback that only people present during the tallying procedure can verify that the votes cast are counted correctly.

There are several electronic Voting (eVoting) schemes that support a remotely verifiable tallying process using a public bulletin board. More precisely, during the vote casting process each voter receives a receipt containing some information, in most schemes the own vote in encoded form. After the polls closed all encoded votes contained in the ballot box are published on the bulletin board and each voter can verify that the own vote has been recorded as cast using his/her receipt (*individual verifiability*). In addition, during the tallying process some audit data is published that allows anybody to check that all votes recorded have been tallied correctly (*universal verifiability*).

In particular, in this scenario homomorphic signatures would allow to verifiably tally votes that have been authenticated leading to an authenticated result. Note that here two cases need to be distinguished. In the first case all votes are signed using the same global, but private, election signing key. In the second case several private keys are used to sign individual votes or set of votes. An example for the first case are poll-site voting schemes where the voters cast their vote in a polling station using official election hardware, e.g. by casting votes using voting machines or by scanning filled out ballots. Here a possible application for homomorphic signatures is that the device in question signs the digitally recorded ballots using a global election signing key. Thus, all votes published on the bulletin board are authenticated and an election outcome signed with this election key can be computed. The second case occurs, for instance, when the votes are cast remotely. Here each voter submits his/her own vote and a possible application for homomorphic signatures is that these votes are signed by his/her individual signing key. In this case tallying the votes published on the bulletin board lead to an aggregated signature on the election outcome, i.e. an election outcome signed by all voters. While the first case requires regular homomorphic signatures, in the second case homomorphic aggregate signatures are needed. In both cases, the operation performed over the signed votes range from simple additions of plaintext votes (see [20]), to polynomial functions (see [1]), e.g. simple operations on encrypted data, up to arbitrary functions (see [53]). Thus, for this use case, linearly, polynomial, fully, and aggregate homomorphic signatures are of interest.

With respect to the hardness assumption, it is sufficient that the signature schemes used are based on the classical problems of Integer Factorization and Discrete Logarithm Problem. Indeed, the election result is made public as soon as the counting is completed. Therefore, there is no need for long-term protection of authenticity. Although it is preferable to determine the election outcome as fast as possible, efficiency is in general a less critical aspect for the tallying process. Indeed, computationally powerful devices (e.g. laptops, voting machines) are usually employed. For the same reasons, having a succinct signature is also less important. In many schemes the signed votes are not published before the poll is closed and the voters do not receive any feedback whether the signatures are correct or not. This is for instance the case when manipulated hardware is used. In this case it would not be possible for an adversary to submit a second set of signed votes whether the signatures of the first set could not be successfully forged. The hardware would be replaced and the voting process would be repeated. Therefore, it is sufficient for the scheme to be secure against the weak adversary. However, there might be other schemes where the signature to the data cast is verified directly. If feedback is given to the voter during vote casting the signature must be secure against the strong adversary. With respect to privacy, it is sufficient for a homomorphic signature scheme to achieve the weak context hiding level. In fact it is well known that the set of possible messages are votes and that the admissible functions correspond to the election methods. If only encrypted votes are signed, then privacy is not needed at all.

Table 5 summarizes the existing homomorphic signature schemes that might be of interest for electronic voting. Regarding linear functions, the scheme proposed in [50] and the ones discussed in [26] fulfill our requirements. The latter publication the schemes are even secure against the strong adversary. Since the maximum number of voters is known a priori, we consider as a valid option also the scheme defined in [12], where the number of admissible linear combinations is fixed *a priori*. These schemes have to be taken into account when the votes are not encrypted, i.e. when privacy is needed. When this is not the case, i.e. the votes are encrypted, also the scheme discussed in [17] is suitable.

With respect to the schemes supporting polynomial functions, none of the approaches provide weak privacy. Therefore, none of them can be considered in case the votes are not encrypted. However, the scheme described in [18] fits all the other requirements, included the strong adversary, and can be used when only encrypted data are signed. Note that the scheme relies on multilinear maps. Therefore it will actually become a promising scheme for electronic voting, once such maps will be practically implemented.

Furthermore, there is a fully homomorphic scheme [32] that satisfies the minimal requirements, achieves weakly context hiding privacy, and is even assumed to be secure against quantum computers. However, it can cope with the weak adversary only and efficiency depends on the election method addressed and needs further analysis.

In addition, the linearly homomorphic aggregate scheme presented in [36] fits the minimal requirements, privacy included. Except for privacy also the scheme proposed in [59] fits the minimal requirements and can therefore be employed in voting schemes where the votes are processed in encrypted form. Both schemes are also secure against the strong adversary.

Summarizing, one can say that for voting schemes with a simple election method (the addition of votes) and where the votes are not necessarily encrypted, homomorphic signatures and aggregate signature are available and there is also a promising fully homomorphic signature scheme. On the other hand, for more complex voting schemes where the votes are even encrypted, a scheme supporting polynomial function is available too. However, in order to address more complex election methods even when the votes are not encrypted, more research is needed if the signature scheme has to support polynomial or arbitrary tallying functions. Furthermore, efficiency is not well defined and more precise analyses in this regard are needed. Note that although this is not a very critical property for this use case, during elections a huge amount of data needs to be processed and the election outcome should be available in reasonable time.

**Table 5.** Suitable Signature Schemes for eVoting

| Signatures: | type | **Hard. Ass** | **Efficiency** | **Size** | **Adversary** | **Privacy** | **Model** |
|---|---|---|---|---|---|---|---|
| [**50**] | Linearly | Lattice | good | constant in $N$ | weak | weak | ROM |
| [**12**] | Linearly | Lattice | $\emptyset$ | short | weak | weak | ROM |
| [**26**] | Linearly | DHP/RSA | $\leq$ [17] | $\geq$ [17] | strong | weak | Standard |
| [**17**] | Linearly | RSA | good | $\leq$ [16] | weak | $\emptyset$ | Standard |
| [**18**] | Polynomial | DHP | good | $\mathcal{O}\left(d^3 + d^2 \log N\right)$ | strong | $\emptyset$ | Standard |
| [**32**] | Fully | Lattice | depends on $f$ | polynomial in depth | weak | weak | ROM |
| [**59**] | Aggregate | Lattice | good | constant in $N$ | strong | $\emptyset$ | ROM |
| [**36**] | Aggregate | Lattice | good | constant in $N$ | strong | weak | ROM |

## 6.2 Smart Grids

Smart grids allow to introduce intelligent electricity generation, load balancing, resource allocation, and dynamic pricing on the basis of real-time power consumptions. The drawback of this new technique is that the collected data allows to generate profiles of the energy consumers. Thus, in order to preserve data privacy of individual households, the so called in-network aggregation is performed. More precisely, the measured data is routed through a set of smart meters where each smart meter aggregates its input. Thus, the result reported to the supplier only provides information about a district but hides the fine-grained individual metering data. To prevent that the meters on the route can see the intermediate results the measurements are encrypted by the smart meters using a homomorphic encryption scheme and aggregated using the homomorphic property. Besides homomorphic encryption also digital signatures provide important functionalities for this use case. They protect against unintentional errors and prevent adversaries from altering messages. However, to be compatible with privacy-preserving in-network data aggregation, signatures with homomorphic properties are needed. They can be used to sign the encrypted metering data and be aggregated along with the corresponding ciphertexts at each intermediate node ([41], [55]). This allows the energy supplier to verify the correctness of the aggregation by checking the consistency between the aggregation result and the aggregation signature.

As we already mentioned, in the framework of smart grids only encrypted data are signed. Furthermore, if each smart meter on the aggregation route makes use of the same private key in the signing procedure, then linearly homomorphic signature schemes and homomorphic signature schemes for polynomial functions are taken into account. More precisely, if in the employed homomorphic encryption scheme the ciphertexts are added, then a linearly homomorphic signature schemes is needed. If instead the ciphertexts are multiplied, then a homomorphic signature scheme supporting polynomial functions has to be employed. On the other hand, if the secret keys differ for each smart meter, then homomorphic aggregate signatures are the suitable ones.

In this context, it is sufficient for a homomorphic signature scheme to be based either on the Discrete Logarithm Problem, or on the Integer Factorization Problem. Indeed there is no need for a long-term storage of the consumptions. However, the signature schemes should provide high performances in terms of efficiency. In fact the power consumption needs to be reported in real-time and therefore signature generation must be fast. In addition, the data are computed and aggregated by the smart meters and they have only restricted resources. Thus, the signatures' size should not be large. The homomorphic signature schemes should also cope with the strong adversary. Indeed, since there is the possibility to resent rejected data, the attacker can perform queries multiple times. For smart metering, privacy is not an issue since only encrypted data are signed. Thus, there are no constraints regarding the context hiding level.

Table 6 summarizes the existing homomorphic signature schemes that might me of interest for smart grids. None of the currently existing linearly homomorphic signature schemes satisfy the minimal requirements for smart grids discussed above. Indeed, the most promising options are the one proposed in [17] and the ones proposed in [26]. The first scheme is efficient and the signature size is not very large. However, it doesn't achieve the desired safety level: it can cope only with the weak adversary. For the schemes proposed in [26] it is the other way round. They provide security even against the strong adversary, but the efficiency and the signature's size are not optimal. Thus, an interesting future work for the smart grid use case would be designing a linearly homomorphic signature scheme merging together the pros of the schemes described in [17] and [26]. Furthermore, as a future work, a more precise comparison between those two schemes should be done in terms of efficiency.

As regards the homomorphic signatures for polynomial functions, the scheme proposed in [18] seems to fit the minimal requirements. Indeed the efficiency is claimed to be good. In addition, the signature's length depends logarithmically on the data set size, that is, the signature is succinct. Finally, the signature scheme is secure even against the strong adversary. However, we recall that this scheme uses an ideal graded encoding scheme, which to our knowledge doesn't exist so far.

Also both of the currently existing homomorphic aggregate signatures ([59], [36]) satisfy these requirements. In fact the efficiency is claimed to be good, even though the hardness assumptions are based on lattices. The schemes can cope with the strong adversary and the signature's size obtained is constant with respect to the data set size. However, also for the homomorphic signature schemes supporting polynomial functions and the homomorphic aggregate ones, a deeper insight regarding efficiency is recommended. In fact, numerical results are useful to have a concrete taste of the real performances of the schemes taken into account before using them in practice.

**Table 6.** Suitable Signature Schemes for Smart Grids

| Signatures: | type | **Hard. Ass** | **Efficiency** | **Size** | **Adversary** | **Privacy** | **Model** |
|---|---|---|---|---|---|---|---|
| **[18]** | Polynomial | DHP | good | $\mathcal{O}\ (d^3 + d^2 \log N)$ | strong | $\emptyset$ | Standard |
| **[59]** | Aggregate | Lattice | good | constant in $N$ | strong | $\emptyset$ | ROM |
| **[36]** | Aggregate | Lattice | good | constant in $N$ | strong | weak | ROM |

## 6.3 Electronic Health Records

In the last years, there has been an increasing interest in moving to digital health records. Indeed in many European countries recording such data electronically is becoming central in the national health informatics strategies (see [38]) and United Kingdom is one of the most advanced in this process (see [49]). Also in the United States the adoption of electronic health records is becoming more and more widespread [6]. Recording health information in a digital fashion make it more reliable and easier to access by different medical facilities, such as medical practices, hospitals, health insurances, medical institutes, and pharmacies. The data stored can be used for merely consultations, but not only for that: one may want to perform computations over them, such as statistical calculations. In case the data are authenticated by a homomorphic signature scheme, then the above issue can be easily addressed, as discussed in [44]. In fact, let us suppose that a doctor has signed several data regarding its patients. Then, another institution, e.g. a medical institute, can perform a computation on a specific dataset, e.g. measured blood pressures, outputting the final result already authenticated accordingly. This scenario can be extended to an input set signed by several doctors in a hospital, where health records are stored in a common data base, and even to several hospitals. In this case, homomorphic aggregate signatures are required to perform the computations, since the original data are signed by doctors using different secret keys.

Summarizing, in this context linearly homomorphic signatures, homomorphic signatures for polynomial functions, fully homomorphic signatures, and homomorphic aggregate signatures are of interest. In addition, depending for how long a certain data is stored, the schemes to take into account can be either the ones based on classic problems (Integer Factorization and Discrete Logarithm Problem) or the ones based on lattices problems. Due to the sensitive nature of the information involved, the properties that a homomorphic signature scheme has to satisfy are demanding. Indeed, the efficiency of the computation should be high: a lot of information is stored every day and the function to be applied might be expensive. However, it is assumed that the devices employed in this scenario are not computationally weak. Therefore, also a large signature's size can still be accepted in case it is not succinct . The scheme should be safe against the strong adversary, since there is the possibility to repeat queries multiple times. In addition, due to the sensitivity of the information, privacy is an important property. However, the weakly context hiding level of privacy is sufficient: a homomorphic signature scheme achieving such level of privacy already doesn't leak information about the original data set. That is, no one can see for example the real blood pressure values of the patients, and only the result of the computations is revealed.

An interesting linearly homomorphic signature schemes is the one presented in [50] that would have been the most promising one. However, it is not acceptable since it is secure against the weak adversary only.

Regarding the homomorphic schemes supporting polynomial functions, none of them fits the minimal requirements. Indeed either the efficiency is not good enough, the required level of privacy is not provided, or the adversary is the weak one. That is the same situation for the fully homomorphic signature schemes.

However, there is one linearly homomorphic aggregate signature scheme that is suitable with respect to electronic health records. That is the one discussed in [36]. In fact, it provides security against the strong adversary and achieves the weak context hiding level of privacy. In addition, even though it is based on lattices, the efficiency is claimed to be good.

Future works should address the aforementioned drawbacks and provide suitable solutions for linearly homomorphic signature schemes, for homomorphic signature schemes supporting polynomial functions and for the fully ones. In addition, deeper analysis regarding efficiency are desirable, especially for the linearly homomorphic aggregated signature scheme that looks promising for this use case (the one proposed in [36]). Indeed, quantitative (rather than only qualitative) comparisons in terms of efficiency would give a better insight regarding the schemes' performance. Furthermore, the results would show whether the efficiency provided is good enough for the schemes to be used in practice or whether more work needs to be done in this direction.

## 7 Conclusion

In this work a formal definition of the general framework regarding homomorphic signature schemes is provided. Starting from such framework, it is also shown how linearly homomorphic signature schemes, homomorphic signature schemes for polynomial functions, fully homomorphic signature schemes, and homomorphic aggregate signature schemes are derived. Afterwards, the first up-to-dated survey about all the currently existing homomorphic signature schemes is provided, where each scheme is singularly described and analyzed. In addition three interesting use cases for homomorphic signature schemes presented. These are: electronic voting, smart grids, and electronic health records. For each use case the minimal requirements a suitable homomorphic signature scheme should fulfill are identified and the existing homomorphic signature schemes that properly address these requirements are presented. Based on these observations, directions for future work are suggested that would address the faults of the current state of the art.

One of the most important directions for future research with respect to homomorphic signature schemes is efficiency. So far, only partial comparisons have been provided, proposing a qualitative rather then quantitative description in this regard. However, a deep analysis involving all the existing schemes is not available yet, even though it would be a very valuable contribution. Indeed, having a clear insight about efficiency, would provide a better understanding of the schemes' performances in practice. Furthermore, it would show how these schemes behave in real-life situations, such as when they have to run on computationally weak devices or process large amounts of data.

With respect to linearly homomorphic schemes, it would be useful to design approaches providing both a good efficiency level and safety against the strong adversary. Such schemes would be suitable to address the minimal requirements of smart grids. In addition, for electronic health records it would be desirable that linearly homomorphic signature schemes are developed that are expected to be resilient even against quantum computer attacks and that achieve at least weak privacy.

Regarding homomorphic signature schemes supporting polynomial functions, it would be interesting to design a scheme that provides at least weak privacy, so that it can be used in the context of electronic voting. Together with privacy, in order to be applicable for electronic health records, homomorphic signature schemes for polynomial functions need to be developed that are secure against the strong adversary.

With respect to the fully homomorphic signature schemes, an important topic for future work is to build a scheme that copes with the strong adversary and achieves at least weak privacy. This would allow to use it together with electronic health records.

Finally, in this work we looked at the use cases only from a high level point of view. Thus, further research should be done before using homomorphic signature schemes for the mentioned applications. However, the discussion of the use cases also showed that not for each scenario an appropriate signature scheme is available and that extensive efficiency analyses are missing. Furthermore, the introduction of homomorphic signature schemes to several use cases needs further research. In our perspective these are interesting directions for future work and we plan to work on these matters in the future.

## Acknowledgment

This work has been co-funded by the European Union's Horizon 2020 research and innovation program under Grant Agreement No 644962. In addition, this project has received funding from the DFG as part of project Long-Term Secure Archiving within the CRC 1119 CROSSING.

## References

1. Ben Adida and Ronald L Rivest. Scratch & vote: self-contained paper-based cryptographic voting. In *Proceedings of the 5th ACM workshop on Privacy in electronic society*, pages 29–40. ACM, 2006.
2. Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, Brent Waters, et al. Computing on authenticated data. In *Theory of Cryptography*, pages 1–20. Springer, 2012.
3. Nuttapong Attrapadung and Benoît Libert. Homomorphic network coding signatures in the standard model. In *Public Key Cryptography–PKC 2011*, pages 17–34. Springer, 2011.
4. Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. Computing on authenticated data: New privacy definitions and constructions. In *Advances in Cryptology–ASIACRYPT 2012*, pages 367–385. Springer, 2012.
5. Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. Efficient completely context-hiding quotable and linearly homomorphic signatures. In *Public-Key Cryptography–PKC 2013*, pages 386–404. Springer, 2013.
6. David Blumenthal and Marilyn Tavenner. The "meaningful use" regulation for electronic health records. *New England Journal of Medicine*, 363(6):501–504, 2010.
7. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology–EUROCRYPT 2004*, pages 56–73. Springer, 2004.
8. Dan Boneh and Xavier Boyen. Efficient selective identity-based encryption without random oracles. *Journal of Cryptology*, 24(4):659–693, 2011.
9. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology–CRYPTO 2004*, pages 41–55. Springer, 2004.
10. Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. In *Public Key Cryptography–PKC 2009*, pages 68–87. Springer, 2009.
11. Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In *Advances in Cryptology–EUROCRYPT 2011*, pages 149–168. Springer, 2011.
12. Dan Boneh and David Mandell 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.
13. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in cryptology–EUROCRYPT 2003*, pages 416–432. Springer, 2003.
14. Xavier Boyen, Xiong Fan, and Elaine Shi. Adaptively secure fully homomorphic signatures based on lattices. 2014.
15. Dario Catalano. Homomorphic signatures and message authentication codes. In *Security and Cryptography for Networks*, pages 514–519. Springer, 2014.
16. Dario Catalano, Dario Fiore, and Bogdan Warinschi. Adaptive pseudo-free groups and applications. In *Advances in Cryptology–EUROCRYPT 2011*, pages 207–223. Springer, 2011.
17. Dario Catalano, Dario Fiore, and Bogdan Warinschi. Efficient network coding signatures in the standard model. In *Public Key Cryptography–PKC 2012*, pages 680–696. Springer, 2012.
18. Dario Catalano, Dario Fiore, and Bogdan Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In *Advances in Cryptology–CRYPTO 2014*, pages 371–389. Springer, 2014.
19. Denis Charles, Kamal Jain, and Kristin Lauter. Signatures for network coding. *International Journal of Information and Coding Theory*, 1(1):3–14, 2009.

20. David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan Sherman, and Poorvi Vora. Scantegrity: End-to-end voter-verifiable optical-scan voting. *Security & Privacy, IEEE*, 6(3):40–46, 2008.
21. Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. Technical report, Cryptology ePrint Archive, Report 2015/162, 2015. http://eprint. iacr. org, 2015.
22. Ronald Cramer and Victor Shoup. Signature schemes based on the strong rsa assumption. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):161–185, 2000.
23. László Czap and István Vajda. Signatures for multisource network coding. Technical report, ArXiv, 2010.
24. Jing Dong, Reza Curtmola, and Cristina Nita-Rotaru. Practical defenses against pollution attacks in wireless network coding. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):7, 2011.
25. Caroline Fontaine and Fabien Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security*, 2007:15, 2007.
26. David Mandell Freeman. Improved security for linearly homomorphic signatures: A generic framework. In *Public Key Cryptography–PKC 2012*, pages 697–714. Springer, 2012.
27. Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology–EUROCRYPT 1999*, pages 123–139. Springer, 1999.
28. Rosario Gennaro, Jonathan Katz, Hugo Krawczyk, and Tal Rabin. Secure network coding over the integers. In *Public Key Cryptography–PKC 2010*, pages 142–160. Springer, 2010.
29. Craig Gentry, Chris Peikert, and Vinod 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.
30. Craig Gentry, Chris Peikert, and Vinod 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.
31. Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
32. Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. Cryptology ePrint Archive, Report 2014/897, 2014. `http://eprint.iacr.org/`.
33. Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In *Advances in Cryptology–ASIACRYPT 2014*, pages 491–511. Springer, 2014.
34. Ryo Hiromasa, Yoshifumi Manabe, and Tatsuaki Okamoto. Homomorphic signatures for polynomial functions with shorter signatures. In *The 30th Symposium on Cryptography and Information Security, Kyoto*. 2013.
35. Susan Hohenberger and Brent Waters. Short and stateless signatures from the rsa assumption. In *Advances in Cryptology–CRYPTO 2009*, pages 654–670. Springer, 2009.
36. Zhengjun Jing. An efficient homomorphic aggregate signature scheme based on lattice. *Mathematical Problems in Engineering*, 2014.
37. Robert Johnson, David Molnar, Dawn Song, and David Wagner. Homomorphic signature schemes. In *Topics in Cryptology?CT-RSA 2002*, pages 244–262. Springer, 2002.
38. Dipak Kalra and David Ingram. Electronic health records. In *Information technology solutions for healthcare*, pages 135–181. Springer, 2006.
39. Jonathan Katz and Brent Waters. Compact signatures for network coding, 2008.
40. Seung-Hoon Lee, Mario Gerla, Hugo Krawczyk, Kang-Won Lee, and Elizabeth A Quaglia. Performance evaluation of secure network coding using homomorphic signature. In *Network Coding (NetCod), 2011 International Symposium on*, pages 1–6. IEEE, 2011.
41. Fengjun Li and Bo Luo. Preserving data integrity for smart grid data aggregation. In *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*, pages 366–371. IEEE, 2012.

42. Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Linearly homomorphic structure-preserving signatures and their applications. In *Advances in Cryptology–CRYPTO 2013*, pages 289–307. Springer, 2013.
43. Vadim Lyubashevsky and Daniele Micciancio. Asymptotically efficient lattice-based digital signatures. In *Theory of Cryptography*, pages 37–54. Springer, 2008.
44. Eleftheria Makri, Maarten H Everts, Sebastiaan Hoogh, Andreas Peter, Harm Akker, Pieter Hartel, and Willem Jonker. Privacy-preserving verification of clinical research. 2014.
45. Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.
46. David Molnar. *Homomorphic signature schemes*. PhD thesis, Citeseer, 2003.
47. Ronald L Rivest. On the notion of pseudo-free groups. In *Theory of cryptography*, pages 505–521. Springer, 2004.
48. Ronald L Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
49. Ann Robertson, Kathrin Cresswell, Amirhossein Takian, Dimitra Petrakaki, Sarah Crowe, Tony Cornford, Nicholas Barber, Anthony Avery, Bernard Fernando, Ann Jacklin, et al. Implementation and adoption of nationwide electronic health records in secondary care in england: qualitative analysis of interim results from a prospective national evaluation. *BMJ*, 341, 2010.
50. FengHe Wang, YuPu Hu, and BaoCang Wang. Lattice-based linearly homomorphic signature scheme over binary field. *Science China Information Sciences*, 56(11):1–9, 2013.
51. Yongge Wang. Insecure" provably secure network coding" and homomorphic authentication schemes for network coding. *IACR Cryptology ePrint Archive*, 2010:60, 2010.
52. Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology–EUROCRYPT 2005*, pages 114–127. Springer, 2005.
53. Zhe Xia, Chris Culnane, James Heather, Hugo Jonker, Peter YA Ryan, Steve Schneider, and Sriramkrishnan Srinivasan. Versatile prêt à voter: Handling multiple election methods with a unified interface. In *Progress in Cryptology-INDOCRYPT 2010*, pages 98–114. Springer, 2010.
54. Wenjie Yan, Mingxi Yang, Layuan Li, and Huajing Fang. Short signature scheme for multi-source network coding. *Computer Communications*, 35(3):344–351, 2012.
55. Lei Yang and Fengjun Li. Detecting false data injection in smart grid in-network aggregation. In *Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on*, pages 408–413. IEEE, 2013.
56. Zhen Yu, Yawen Wei, Bhuvaneswari Ramkumar, and Yong 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.
57. Aaram Yun, Jung Hee Cheon, and Yongdae Kim. On homomorphic signatures for network coding. *IEEE Transactions on Computers*, (9):1295–1296, 2010.
58. Ning Zhang. Signatures for network coding. 2010.
59. Peng Zhang, Jianping Yu, and Ting Wang. A homomorphic aggregate signature scheme based on lattice. *Chinese Journal of Electronics*, 21(4):701–704, 2012.
60. Fang Zhao, Ton Kalker, Muriel Médard, and Keesook J Han. Signatures for content distribution with network coding. In *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pages 556–560. IEEE, 2007.