

Chapter 2

Homomorphic Signature Schemes

Abstract In this chapter two types of signature schemes satisfying homomorphic properties are presented. In the first section a description of the homomorphic signature schemes suitable in the single-user scenario is provided. In the second section the homomorphic signature schemes that support the multi-user case are presented.

2.1 Homomorphic Signature Schemes for the Single-User Scenario

There are three different types of homomorphic signature schemes for the single-user case. In fact, the whole set of homomorphic signatures can be divided according to the admissible functions each scheme supports. Specifically, we can distinguish schemes providing:

- linear functions, so called linearly homomorphic signature schemes;
- polynomial functions, so called homomorphic signature schemes for polynomial functions;
- arbitrary functions, so called fully homomorphic signature schemes.

The signatures are presented in the same order as listed above. That is, they are discussed with respect to a set of possible admissible functions which is less and less restrictive. For each section, the differences with respect to the general definition of homomorphic signatures are highlighted and the evolution from linearly up to fully homomorphic signature schemes is shown.

2.1.1 *Linearly Homomorphic Signature Schemes*

A linear homomorphic signature scheme [45] allows to perform linear functions over signed messages. With respect to the general definition of homomorphic signature schemes, 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. $v = (a_1, a_2, \dots, a_N)$ where $a_i \in \mathbb{F}_p$.
- If we consider the vectors v_1, v_2, \dots, v_N , then the set \mathcal{F} of admissible functions $f \in \mathcal{F}$ are all possible linear combinations in the \mathbb{F}_p -linear span of v_1, v_2, \dots, v_N .

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

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

The definition of linearly homomorphic signatures follows [18].

Definition 2.1. A *linearly homomorphic signature scheme* is a tuple of the following probabilistic, polynomial-time algorithms:

- *Set*($1^\lambda, N$). It takes as input a security parameter λ in unary and an integer $N > 0$. It outputs a secret key sk , the respective public key pk , 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 $f : \mathbb{F}_p^N \rightarrow \mathbb{F}_p^N$.
- *Sig*(sk, τ, v, i). It takes as input a secret key sk , a tag $\tau \in \{0, 1\}^\lambda$, a vector $v \in \mathbb{F}_p^N$, and an index $i \in \{1, 2, \dots, N\}$. It outputs a signature $\sigma \in \mathbb{F}_p^N$, computed using the secret key sk , which is the signature for the i -th message v of the data set tagged by τ .
- *Vrf*(pk, τ, v, σ, f). It takes as input a public key pk , 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 ‘1’ if σ is a valid signature for the vector v . Such vector v is output of the function f over the data set tagged by τ , whose messages are signed using the public key pk . It outputs ‘0’ otherwise.
- *Eval*($pk, \tau, f, \vec{\sigma}$). It takes as input a public key pk , a tag $\tau \in \{0, 1\}^\lambda$, a function $f \in \mathcal{F}$, and a tuple of signatures $\vec{\sigma} \in \mathbb{F}_p^N$. It outputs a signature $\sigma' = \sum_{i=1}^N c_i \sigma_i \in \mathbb{F}_p^N$ for a function $f \in \mathcal{F}$ over the (tuple of) signatures $\vec{\sigma} \in \mathbb{F}_p^N$. Such tuple $\vec{\sigma}$ corresponds to the signatures on the messages within the data set labeled by tag $\tau \in \{0, 1\}^\lambda$.

For the *correctness*, we refer to Definition 1.11, where the message $m \in \mathcal{M}$ is a vector $v \in \mathbb{F}_p^N$ and $f(\vec{m}) = \sum_{i=1}^N c_i v_i$.

2.1.2 Homomorphic Signature Schemes for Polynomial Functions

A homomorphic signature scheme for polynomial functions is a signature scheme that allows to compute polynomial functions over signed messages. The first of such schemes is proposed in [14], where the polynomials are multivariates of bounded degree. It can be seen as a generalization of linearly homomorphic schemes, since in the linearly case the set of admissible functions is a polynomial of degree one. The following definition of homomorphic signatures for polynomial functions is a generalization of the original one presented in [14].

As described in [14], the general framework of such signatures is composed of the following elements.

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

The homomorphic property for this type of signature schemes follows. Given one signature *per* message m_1, m_2, \dots, m_N , anyone can compute a signature for the polynomial $f(\vec{m}) = \sum_{j=1}^{\ell} c_j Y_j(\vec{m})$ where:

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

Definition 2.2. A homomorphic signature scheme for polynomial functions is a tuple of the following probabilistic, polynomial-time algorithms:

- *Set*($1^\lambda, N$). It takes as input a security parameter λ in unary and an integer $N > 0$. It outputs a secret key sk , the respective public key pk , a prime number p , and a monic irreducible polynomial $F(x) \in \mathbb{Z}[x]$ of degree d . The public key determines the space of messages \mathbb{F}_p , the space of signatures R , and the set $\mathcal{F} \subset \mathbb{F}_p[x_1, \dots, x_N]$ of admissible functions, where $y = \text{poly}(\lambda)$ and $d = \mathcal{O}(1)$.
- *Sig*(sk, τ, m, i). It takes as input a secret key sk , a tag $\tau \in \{0, 1\}^\lambda$, a message $m \in \mathbb{F}_p$, and an index $i \in \{1, 2, \dots, N\}$. It outputs a signature $\sigma \in R$, computed using the secret key sk , which is the signature for the i -th message m of the data set tagged by τ .
- *Vrf*(pk, τ, m, σ, f). It takes as input a public key pk , 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 ‘1’ if σ is the valid

signature for the message m . Such message m is the output of the function f over the data set tagged by τ , whose messages are signed using the public key pk . It outputs '0' otherwise.

- $Eval(pk, \tau, f, \vec{\sigma})$. It takes as input a public key pk , a tag $\tau \in \{0, 1\}^\lambda$, a function $f \in \mathcal{F}$, a tuple of signatures $\vec{\sigma} \in R$. It outputs a signature $\sigma' = f(\vec{\sigma}) \in R$ for a function $f \in \mathcal{F}$ over the (tuple of) signatures $\vec{\sigma} \in R$. Such tuple $\vec{\sigma}$ corresponds to the signatures on the messages within the data set labeled by tag $\tau \in \{0, 1\}^\lambda$.

For the *correctness*, we refer to Definition 1.11, where $f(\vec{m}) = \sum_{j=1}^{\ell} c_j Y_j(\vec{m})$.

Remark 2.1. Note that in [14], the computation of $f \in \mathbb{F}_p[x_1, x_2, \dots, x_N]$ over the tuple of signatures $\vec{\sigma}$ is actually performed in two steps. First, f is lifted to a function, $\hat{f} \in \mathbb{Z}[x_1, x_2, \dots, x_N]$ defined as $\hat{f} := \sum_{j=1}^{\ell} c_j Y_j(x_1, x_2, \dots, x_N)$, where c_1, c_2, \dots, c_N are integer coefficients. Then, σ' is computed as the output of $\hat{f}(\vec{\sigma})$.

2.1.3 Fully Homomorphic Signatures

Using fully homomorphic signature schemes there are no restrictions with respect to the operations that can be performed on the signed messages. Now, being allowed to use both $+$ and \times over a field \mathbb{F}_p , it is possible to evaluate any function. Such function is now described by a *circuit* C with a certain size and a certain depth d . We do not propose here the definition of a fully homomorphic signature scheme, since it is almost the same as for a general homomorphic signature (Definition 1.10). However, the few variations to take into account are discussed.

- The function is seen as a circuit, which is denoted as $C : \mathcal{M}^N \rightarrow \mathcal{M}$.
- Instead of the set of admissible functions, a circuit family \mathcal{C} is employed.
- The algorithm *Setup* outputs the secret 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. $P(m_1, \dots, m_N) = m_i$. This means that the correctness must also hold for single-message signatures (see [19]).

For the *correctness*, due to the generality of the function that a fully homomorphic scheme supports, we refer directly to Definition 1.11, where the description of such f covers all types of functions.

2.2 Homomorphic Signature Schemes for the Multi-Users Scenario

The homomorphic signature schemes presented so far are suitable for the single-user scenario. There is only one signer owning the secret key that generates authenticated

messages. When there are multiple users involved in the signing process, then homomorphic signature schemes supporting the multi-users case are needed. In this scenario we want to perform operations on signatures on messages signed by different users, each of them with its own private key. Among these schemes, the following two types of multi-users signature schemes can be distinguished:

- multi sources homomorphic signature schemes;
- homomorphic aggregate signature schemes.

2.2.1 Multiple Sources Homomorphic Signature Schemes

As for the linearly homomorphic signature schemes, the multiple sources homomorphic schemes were at first introduced by Agrawal et al. in [3] to address pollution attacks within the network coding framework (for a precise explanation we refer to [18]). Multi sources network coding refers to the situation where several sources transmit data, instead of a single one. For this reason in literature the latter schemes are called “multiple sources network coding signature schemes”. Though, we refer to them omitting the link to network coding, as such signature schemes allow for supporting multiple users, and not only multiple sources.

As the homomorphic signature schemes for a single user, the multiple sources homomorphic ones are defined over a message space \mathcal{M} , a space of signed messages \mathcal{S} , both of them equipped with an operation, a space of secret keys \mathcal{K} , and a space of public keys \mathcal{K}' . They still provide a setup, signing, verifying, and evaluating algorithm. All of them are probabilistic, polynomial-time algorithms, except for the verification algorithm that is deterministic. Though, in order to support multiple users, some differences are introduced in the definition of multiple sources homomorphic signature schemes.

- The algorithm *Set* takes as input a security parameter and a positive integer N , which stands for the maximum number of users the scheme can support. In addition, the algorithm *Set* returns N secret-public key pairs.
- The algorithm *Sig* takes as input a secret key, a message, and an index $i \in \{1, 2, \dots, N\}$, in order to specify the user i together with its secret key sk_i and message m_i .
- The algorithm *Vrf* takes as input a string of N public keys pk_1, pk_2, \dots, pk_N (indicated by \vec{pk}), one for each signer, a tag, a message, a signature, and an admissible function.
- The algorithm *Eval* takes as input a string of N public keys \vec{pk} , a tag, an admissible function, and a string of N signatures $\sigma_1, \sigma_2, \dots, \sigma_N$ (indicated by $\vec{\sigma}$), one for each signer.

In the following we provide a more precise definition of multiple sources homomorphic signature schemes. The original definition provided by Agrawal et al.

in [3] is specific for multiple sources network coding. We adapt the definition to the more general multi-users scenario.

Definition 2.3. A *multiple sources homomorphic signature scheme* is a tuple of the following probabilistic, polynomial-time algorithms:

- $Set(1^\lambda, N)$. It takes as input a security parameter λ in unary and an integer $N > 0$. It outputs N pairs (sk_i, pk_i) of secret and public keys, one for each user i . The public key determines the space of messages \mathcal{M} , the space of signatures \mathcal{Y} , and the set \mathcal{F} of admissible functions $f : \mathcal{M}^N \rightarrow \mathcal{M}$.
- $Sig(sk, \tau, m, i)$. It takes as input a secret key sk , a tag $\tau \in \{0, 1\}^\lambda$, a message $m \in \mathcal{M}$, and an index $i \in \{1, 2, \dots, N\}$. It outputs a signature $\sigma \in \mathcal{Y}$, computed using the i -th secret key sk , which is the signature for the i -th message m of the data set tagged by τ .
- $Vrf(\vec{pk}, \tau, m, \sigma, f)$. It takes as input a public keys' string \vec{pk} , 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 '1' if σ is a valid signature for the message m . Such message m is the output of the function f over the data set tagged by τ , whose messages are signed using the public keys in the string \vec{pk} . It outputs '0' otherwise.
- $Eval(\vec{pk}, \tau, f, \vec{\sigma})$. It takes as input a public keys' string \vec{pk} , a tag $\tau \in \{0, 1\}^\lambda$, a function $f \in \mathcal{F}$ and a tuple of signatures $\vec{\sigma} \in \mathcal{Y}^N$. It outputs a signature $\sigma' \in \mathcal{Y}$ on the output of a function $f \in \mathcal{F}$ over the (tuple of) signatures $\vec{\sigma} \in \mathcal{Y}^N$. Such tuple $\vec{\sigma}$ corresponds to the signatures on the messages within the data set labeled by tag $\tau \in \{0, 1\}^\lambda$.

The definition *correctness* takes into account both the homomorphic property and the fact that multiple public keys may be involved in the verification process. When a single signature has to be verified the algorithm Vrf takes as input one public key only. As for the homomorphic signature schemes in the single-user case, the admissible function f is meant as a projection from the data set to the message in question and will not be specified.

Definition 2.4. A homomorphic aggregate signature is *correct* if for each of the N secret-public key pair (sk, pk) output of the algorithm $Set(1^\lambda, N)$, the following conditions are valid:

- (1) For all $\tau \in \{0, 1\}^\lambda$ and $i \in \{1, 2, \dots, N\}$, if σ is the output of $Sig(sk, \tau, m, i)$, then $Vrf(pk, \tau, m, \sigma) = 1$.
- (2) For all $\tau \in \{0, 1\}^\lambda$ and $i \in \{1, 2, \dots, N\}$, for all pairs (m_i, σ_i) and (sk_i, pk_i) with $Vrf(pk_i, \tau, m_i, \sigma_i) = 1$

$$Vrf(\vec{pk}, \tau, f(\vec{m}), \vec{\sigma}, Eval(\vec{pk}, \tau, f, \vec{\sigma})) = 1,$$

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

2.2.1.1 Multiple Sources Linearly Homomorphic Signature Schemes

The existing multiple sources homomorphic signature schemes proposed so far support linear operations over signed messages. We refer to these schemes as *multiple sources linearly homomorphic signature schemes*, even if in literature the adjective “linearly” is omitted, like for the aggregate case.

The definition of multiple sources linearly homomorphic signature schemes is quite similar to the single-user counterpart and can be easily derived from Definition 2.3. In the following, the changes to take into account to allow a multiple sources homomorphic signature scheme to support linear combinations on signatures are described. For a more formal definition we refer to [75].

- The messages space \mathcal{M} and the signatures space \mathcal{S} 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. $v = (a_1, a_2, \dots, a_N)$ where $a_i \in \mathbb{F}_p$.
- 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, \dots, v_N .

2.2.2 Homomorphic Aggregate Signature Schemes

In this section we discuss homomorphic aggregate signature schemes. Aggregate signatures allow to aggregate different signatures on different messages, signed by different users, each of them with its own secret key. This type of signature schemes has been firstly introduced by Boneh et al. in [17]. In the case we want to perform operations on the signatures rather than only aggregating them we need so-called *homomorphic aggregate signature schemes*. In the following, we first discuss the definition and the framework of aggregate signatures are. Then we define homomorphic aggregate signatures and finally describe the schemes supporting linear functions.

2.2.2.1 Aggregate Signature Schemes

An aggregate signature scheme combines multiple signatures into a single one. Assume we have N different messages and their N respective signatures generated by different secret-public key pairs. If the signatures have been generated using an *aggregate signature scheme* it is possible to generate a single signature for all the N messages.

More precisely, suppose that N users u_1, u_2, \dots, u_N want to obtain a signature σ which is an aggregation of the respective signatures $\sigma_1, \sigma_2, \dots, \sigma_N$. Assume in addition that each message-signature pair (m_i, σ_i) has been generated by user u_i , using its own secret-public key pair (sk_i, pk_i) . Then it is possible to aggregate these

N signatures into a single one. This signature should be short, i.e. it should not be longer than the original N signatures. Another important property of such scheme is to be *incremental*. That is, after aggregating N signatures $\sigma_1, \sigma_2, \dots, \sigma_N$ and receiving the signature σ , it is always possible to aggregate σ and a further one σ_{N+1} . This means that we do not have to start the process from the scratch and run another aggregate signature scheme for $\sigma_1, \sigma_2, \dots, \sigma_N, \sigma_{N+1}$. Instead, it is possible to generate the final signature σ' out of σ_{N+1} and σ .

As for any other signature, also the aggregate one is defined over the messages space \mathcal{M} , the signatures space \mathcal{Y} , and the set of secret and public keys \mathcal{K} and \mathcal{K}' , respectively. Furthermore, it defines the standard algorithms *Set*, *Sig*, and *Vrf*.

- The algorithm *Set* chooses the secret key (used in the signing process) and the respective public key (used in the verification process) for each user.
- The algorithm *Sig* takes as input a secret key, a message, and the index $i \in \{1, 2, \dots, N\}$, of user u_i and outputs a signature.
- The algorithm *Vrf* takes as input a message, a signature, and a string of N public keys pk_1, pk_2, \dots, pk'_N (indicated by \vec{pk}), one for each signer and checks the correctness of the signature.

In addition a new algorithm Agg_σ is introduced.

- The algorithm Agg_σ takes as input the signatures $\sigma_1, \sigma_2, \dots, \sigma_N$, aggregate them, and outputs the resulting signature σ .

More precise, aggregate signature schemes are defined as follows.

Definition 2.5. An *aggregate signature scheme* is a tuple of the following probabilistic, polynomial-time algorithms:

- $\text{Set}(1^\lambda)$. It takes as input a security parameter λ in unary. It outputs a secret-public key pair (sk_i, pk_i) for each user i . The public keys determine the space of messages \mathcal{M} and the space of signatures \mathcal{Y} .
- $\text{Sig}(sk, m, i)$. It takes as input a secret key sk , 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 sk .
- $\text{Vrf}(\vec{pk}, m, \sigma)$. It takes as input the public keys' string \vec{pk} , a message $m \in \mathcal{M}$, and a signature $\sigma \in \mathcal{Y}$. It outputs '1' if σ is a valid signature for the message m , signed using the public keys \vec{pk} . It outputs '0' otherwise.
- $\text{Agg}_\sigma(\vec{pk}, \vec{m}, \vec{\sigma})$. It takes as input a public keys' string \vec{pk} , a messages's string $\vec{m} \in \mathcal{M}$, and a signatures' string $\vec{\sigma} \in \mathcal{Y}$. It outputs a signature $\sigma_{\text{agg}} \in \mathcal{Y}$, which is the aggregate signature of the signatures in $\vec{\sigma}$ of the messages in \vec{m} , signed using the public keys in \vec{pk} , 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 for the signature over the aggregated message. In the first case, the algorithm *Vrf* takes as input just one public key pk , that is the one

corresponding to the secret key sk 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 scheme.

Definition 2.6. An aggregate signature scheme is *correct* if for each output (sk, pk) of the algorithm $Set(1^\lambda)$, the following conditions are valid:

- (1) For all $i \in \{1, 2, \dots, N\}$, if σ is the output of $Sig(sk, m, i)$, then $Vrf(pk, m, \sigma) = 1$.
- (2) For all $i \in \{1, 2, \dots, N\}$, if σ_i is the output of $Sig(sk, \tau, m, i)$, then $Vrf(\vec{pk}, \vec{m}, Agg_\sigma(\vec{pk}, \vec{m}, \vec{\sigma})) = 1$.

Remark 2.2. Everything described so far also holds for any arbitrary subset U of the N users, where $0 < |U| < N$.

2.2.2.2 Homomorphic Aggregate Signature Schemes

Homomorphic aggregate signature schemes combine two properties [74]. They are at the same time:

- a signature scheme that aggregates signatures produced by different users (*aggregate signature schemes*);
- a signature scheme that performs computations on signatures using an admissible function (*homomorphic signature schemes*).

In the following, the homomorphic property is added (that is, the possibility to compute on authenticated data) to the definition of aggregate signature schemes. Some differences have to be taken into account.

- The algorithm Set takes as input a security parameter and, in addition, an integer N , which stands for the maximum number of users the scheme can support. Therefore, the parameter N has to be decided a priori and this means that the incremental property of aggregate signatures is lost.
- The algorithm Agg_σ takes as input a public keys' string, a messages' string, and a signatures' string. In addition, it takes as input a tag $\tau \in \{0, 1\}^\lambda$ and an admissible function $f \in \mathcal{F}$. Instead of simply aggregating the signatures also function f is applied.
- 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 performs f on the messages m_1, m_2, \dots, m_N .
- 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 the admissible function $f \in \mathcal{F}$.

The following definition formally addresses the modifications discussed above [74].

Definition 2.7. A *homomorphic aggregate signature scheme* is a tuple of the following probabilistic, polynomial-time algorithms:

- $Set(1^\lambda, N)$. It takes as input a security parameter λ in unary and an integer $N > 0$. It outputs N pairs (sk_i, pk_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 $f : \mathcal{M}^N \rightarrow \mathcal{M}$.
- $Sig(sk, \tau, m, i)$. It takes as input a secret key sk , a tag $\tau \in \{0, 1\}^\lambda$, a message $m \in \mathcal{M}$, and an index $i \in \{1, 2, \dots, N\}$. It outputs a signature $\sigma \in \mathcal{Y}$, computed using the i -th secret key sk , which is the signature for the i -th message m of the data set tagged by τ .
- $Vrf(\vec{pk}, \tau, m, \sigma, f)$. It takes as input a public keys' string \vec{pk} , 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 '1' if σ is a valid signature for the message m , signed using the public keys \vec{pk} , output of the function f over the data set tagged by τ . It outputs '0' otherwise.
- $Agg_m(\vec{pk}, \tau, \vec{m}, f)$. It takes as input a public keys' string \vec{pk} , a tag $\tau \in \{0, 1\}^\lambda$, a messages' string $\vec{m} \in \mathcal{M}$, and an admissible function $f \in \mathcal{F}$. It outputs an aggregated message $m_{Agg} \in \mathcal{M}$ by applying function f on the messages \vec{m} in the data set labeled by tag τ , coming from the users with the public keys \vec{pk} .
- $Agg_\sigma(\vec{pk}, \tau, \vec{\sigma}, f)$. It takes as input a public keys' string \vec{pk} , a tag $\tau \in \{0, 1\}^\lambda$, a signatures' string $\vec{\sigma} \in \mathcal{M}$, and an admissible function $f \in \mathcal{F}$. It outputs an aggregated signature σ_{Agg} by applying function f on the signatures $\vec{\sigma}$. The signatures in the string $\vec{\sigma}$ are the signatures on the messages within the data set labeled by tag τ , coming from the users with the public keys \vec{pk} .

The *correctness* definition takes into account the new algorithm Agg_m and the introduction of the homomorphic property. When a single signature is verified, as for the aggregate signature schemes, the algorithm Vrf takes as input just one public key. In addition, as for the homomorphic signature schemes, the admissible function f is meant as a projection from the data set to the message in question.

Definition 2.8. A homomorphic aggregate signature is *correct* if for each output of secret-public key pair (sk, pk) of the algorithm $Set(1^\lambda, N)$, the following conditions are valid:

- (1) For all $\tau \in \{0, 1\}^\lambda$ and $i \in \{1, 2, \dots, N\}$, if σ is the output of $Sig(sk, \tau, m, i)$, then $Vrf(pk, \tau, m, \sigma) = 1$.
- (2) For all $i \in \{1, 2, \dots, N\}$, if σ_i is the output of $Sig(sk, \tau, m, i)$, then

$$Vrf(\vec{pk}, \tau, Agg_m(\vec{pk}, \tau, \vec{m}, f), Agg_\sigma(\vec{pk}, \tau, \vec{\sigma}, f), f) = 1.$$

2.2.2.3 Linearly Homomorphic Aggregate Signatures

The homomorphic aggregate signature schemes present in literature so far ([45] and [74]) are the linearly ones. That is, the computation supported is a linear combination of different messages m_1, m_2, \dots, m_N coming from different users. Such computation is then reflected on the signatures counterpart. In fact, the final signature σ' joins together the signatures $\sigma_1, \sigma_2, \dots, \sigma_N$, according to the same linear combinations as the one 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{S} 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. $v = (a_1, a_2, \dots, a_N)$ where $a_i \in \mathbb{F}_p$.
- If the vectors v_1, v_2, \dots, v_N are a basis for \mathbb{F}_p^N , then the set 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, \dots, v_N .

For a formal definition of linearly homomorphic aggregate signature schemes Definition 2.7 can be adapted. More precisely, the admissible functions are of the form $f = \sum_{i=1}^N c_i v_i$ with respect to the messages and of the form $f = \sum_{i=1}^N c_i \sigma_i$ for the signatures. The same holds for correctness.

Remark 2.3. A formal definition of linearly homomorphic aggregate signature schemes is provided in [74] and [45]. Though, note that in literature the linearly homomorphic aggregate signatures are called homomorphic aggregate signatures. Indeed, the unique examples available so far allow for linear combinations only, that is why “linearly” is omitted. However, we think that it is important 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.



<http://www.springer.com/978-3-319-32114-1>

Homomorphic Signature Schemes

A Survey

Traverso, G.; Demirel, D.; Buchmann, J.

2016, XI, 64 p., Softcover

ISBN: 978-3-319-32114-1