Implementation and Performance Analysis of Homomorphic Signature Schemes

Davide Carnemolla[1 [0009−0001−2575−0874]], Dario Catalano[1 [0000−0001−9677−944X]], Mario Di Raimondo[1 [0000−0002−5531−1098]], and Federico Savasta[2 [0009−0005−4551−1562]]

[1] University of Catania, Italy
{davide.carnemolla,dario.catalano,mario.diraimondo}@unict.it
[2] federico.savasta94@gmail.com

Abstract. Homomorphic signatures allow to validate computation on signed data. Alice, holding a dataset, $\{m_1, \ldots, m_t\}$ uses her secret key sk to sign these data and stores the authenticated dataset on a remote server. The server can later (publicly) compute $m = f(m_1, ..., m_t)$ together with a signature σ certifying that m is indeed the correct output of the computation f. Over the last fifteen years, the problem of realizing homomorphic signatures has been the focus of numerous research works, with constructions now ranging from very efficient ones supporting linear functions to very expressive ones supporting (up to) arbitrary circuits. In this work we tackle the question of assessing the practicality of schemes belonging to this latter class. Specifically, we implement the GVW lattice based scheme for circuits from STOC 2015 and two, recently proposed, pairings based constructions building from functional commitments. Our experiments show that (both) pairings based schemes outperform GVW on all fronts.

Keywords: Homomorphic Signatures · Implementations.

1 Introduction

Digital Signatures are one of the most fundamental primitives of modern cryptography. Their main security property is unforgeability against chosen message attacks: any efficient attacker $\mathcal{A}$ who has access to signatures on messages of his choice should not be able to produce a signature on a new message. Said differently, this means that signatures should be *non-malleable* in the sense that from a given signature on some message m one should not be able to derive signatures on any other $m' \neq m$. Yet, even if this sounds paradoxical at first, there are scenarios where some form of (controlled) malleability for signatures turns out to be useful. This controlled malleability feature is captured, for example, by the notion of homomorphic signatures, originally proposed by Desmedt [25] and Johnson et al. [31], and later formalized by Boneh and Freeman in [11]. Homomorphic Signatures are the focus of this work.

Homomorphic Signatures. Using homomorphic signatures (HS) some user Alice can use her secret key sk to produce signatures $\sigma_1, \ldots, \sigma_t$ on a set of messages $(m_1, \ldots, m_t)$ (often referred to as a *dataset*) like with ordinary signature scheme. The extra feature of HS, however, is that they come equipped with a (publicly) executable algorithm Eval that takes as input the signatures $\sigma_1, \ldots, \sigma_t$ and some function/program $\mathcal{P}$ and outputs a signature $\sigma_{\mathcal{P}}$ for the message $m = \mathcal{P}(m_1, \ldots, m_t)$. Notice that $\sigma_{\mathcal{P}}$ is not a signature on m alone but rather a signature on m *as output of* $\mathcal{P}$. In this sense, homomorphic signatures implement the requirement of controlled malleability mentioned above. More precisely, on input Alice's public key, $\mathcal{P}, m$ and $\sigma_{\mathcal{P}}$, the verification procedure checks whether m is the correct output of $\mathcal{P}$ on messages previously signed by Alice.

What makes this primitive non trivial to realize are the following features. First, homomorphic signatures should be *succinct*, in the sense that their size should be much smaller than the size of the original dataset[3].

Second, when verifying the output of a computation, the process should not require knowledge of the original dataset, an important property when signing very large datasets that the verifier might not store locally.

Finally, homomorphic signature could be *composable* in the sense that the outputs of previous computations (i.e. signatures obtained by Eval) can be used as inputs for new computations.

Defining security for homomorphic signatures is a much more delicate task[4] as, clearly, HS cannot meet the usual unforgeability notion. Informally, an Homomorphic Signature is secure as long as adversaries knowing only the public verification key, can only produce signatures that either come from the legitimate signer or by running Eval on valid signatures. Slightly more in detail, $\mathcal{A}$ is allowed to see signatures for messages belonging for different datasets and wins if she manages to produce a valid signature, message pair (σ, m) such that either (1) m belongs to some new, previously unseen, dataset (this is called type 1 forgery) or (2) for some previously seen dataset $\Delta = \{m_1, \ldots, m_t\}$, she manages to produce a triplet $(\mathcal{P}, \sigma, m)$, where σ verifies correctly but m is not the correct output of $\mathcal{P}$, i.e. $m \neq \mathcal{P}(m_1, \ldots, m_t)$ (this is called a type 2 forgery).

Over the last fifteen years, the problem of realizing homomorphic signatures under various assumptions and settings has been the focus of several research works. It is fair to say that, at least from a theoretical perspective, nowadays we have a pretty clear understanding of the primitive, with known constructions ranging from very efficient ones for the case of programs expressible as linear functions (e.g. [10]) to very expressive ones supporting up to arbitrary circuits ([30, 7]. Yet, especially for this latter class of schemes, their concrete efficiency is far from clear.

[3] Indeed, without this requirement, HS become trivial to realize: one simply sets $\sigma_{\mathcal{P}} = (\mathcal{P}, (\sigma_1, m_1), \ldots, (\sigma_t, m_t))$

[4] See [19] for a detailed discussion about this

1.1 Our contribution

In this paper we implement three different homomorphic signatures to better asses their concrete efficiency and clarify their practical significance. Specifically, we implement the lattice based HS from [30] (GVW15 from now on) and the pairing based constructions from [20] and [7] (respectively, CTF22 and BCFL23 from now on)[5]. See Table 1 for a brief summary on their main features. These constructions, while concretely very different, share some high level interesting similarities. For instance, they all build from commitments with special properties (GVW15 implicitly builds from homomorphic equivocable commitments, CTF22 and BCFL23 from homomorphic functional commitments). Also, they all admit similar (amortized) efficiency optimizations when verifying the same function on different datasets. These similarities, among other thing, helped make implementation choices allowing fair comparisons among the three schemes.

All the schemes were implemented from scratch in Rust using state of the art libraries (qFALL [38] for lattices and *blst* [40] for pairings). For simplicity all our implementations are single thread and consider the case of a single dataset. Thus, we don't consider optimizations related to efficient verification for multiple datasets: being this a common feature of all the considered schemes, dropping it, has little, if any, impact on our comparisons. Precise details on our implementations and comparisons are given in Sections 5 and 6, here we simply mention that our experiments show that (both) pairing based schemes outperform GVW15 on essentially all fronts. Table 2 reports a sample from our experiments (more details are given in Section 6) for the variance function computed on a vector of $n = 20$ elements: as later discussed in Subsection 5.2, we were forced to use a very low (almost meaningless) security level for GVW15 in order to complete the experiments. Thus, even though there is still room for optimizations/improvements (e.g. multi-thread implementations, ring variants for the lattice scheme), our results suggest that, if one wants efficient homomorphic signatures beyond linear functions, pairing based constructions are the best, currently available, option.

	CFT22	BCFL23	GVW15
security assumption	DHE	n-HiKer[6]	SIS
security notion	strong adapt.	strong adapt.	semi adapt.
class of functions	const. deg. poly	circuits	circuits
amortized verification	✓	✓	✓
multi datasets	✓	✓	✓

Table 1: Features

[5] We stress here that the construction from [20] only supports low degree polynomials whereas the constructions in [30, 7] support arbitrary circuits

	CFT22	BCFL23	GVW15
KeyGen	**0.5** s	5.8 m	2.4 h
Sign	0.2 s	**0.1** s	52.5 h
Eval	44.1 s	7.8 m	**18.7** s
Ver	**0.3** s	5.8 s	6.4 s
signature	39.7 kB	**14.2** kB	29.9 MB
proof	2.1 kB	**1.7** kB	3.9 MB
security level	**100** bits	**100** bits	30 bits

Table 2: Benchmark for variance on $n = 20$ elements

1.2 Other related work

Linearly homomorphic signatures were introduced by Boneh *et al.* in [10] as a tool to prevent pollution attacks in linear network coding. Following this seminal work, many results further explored this notion both in the random oracle (e.g. [28, 12, 17]) and in the standard model (e.g. [3, 21, 27, 22, 5, 18]). In the symmetric setting, the notion of homomorphic MAC was first considered (for the case of linear functions) in [1] and later extended to more general functionalities in [29, 15, 6, 16]. Homomorphic signatures building from commit and prove SNARKs for NP were recently proposed in [26]. Linearly Homomorphic Structure preserving Signatures were introduced in [34] and exploited to construct Simulation Sound Semi Adaptive NIZKs in [35]. Privacy notions for homomorphic signatures were considered in [4].

2 Preliminaries

Notation. In this work we use $\mathbb{N}, \mathbb{Z}, \mathbb{R}$ for the set of natural, integer and real numbers, while $\mathbb{G}$ denotes a group. $\mathbb{Z}_q$ denotes the set of integers mod q if not specified differently. Vectors and matrices are denoted with lowercase and uppercase bold latin letters, respectively, e.g. $\boldsymbol{v}$ and $\mathbf{A}$. λ is the security parameter. With $x \xleftarrow{\$} \mathcal{X}$ we denote the uniform random sampling from a set $\mathcal{X}$, while with $x \leftarrow \mathcal{D}$ a sampling from a distribution or an algorithm $\mathcal{D}$. $\mathsf{poly}(\cdot)$ and $\mathsf{negl}(\cdot)$ denotes the usual polynomial and negligible functions, while $O(\cdot)$ and $\omega(\cdot)$ are the usual big-O and small-ω asymptotic notations. Remaining notation is introduced in the dedicated sections.

[6] The construction is secure under a falsifiable assumption called HintedKernel. We refer the reader to the original paper for the formal definiton.

2.1 Functional Commitments

Intuitively, Functional Commitments (FC) [36] allow a sender to commit to a vector $\boldsymbol{x}$ so that she can later open the commitment to $f(\boldsymbol{x})$ for some function f. We present the formal definition of FC as presented in CTF22, which takes into account also the *compactness* property [33].

Definition 1 (Functional Commitments [20]). *A functional commitment scheme is a tuple of algorithms* $\mathsf{FC} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Ver})$ *with the following syntax and that satisfies correctness and succinctness (or compactness [33]).*

$\mathsf{Setup}(1^\lambda, n, m) \to \mathsf{ck}$ *on input the security parameter* λ *and the vector length* n*, outputs a commitment key* ck*, which defines the message space* $\mathcal{X}$ *and the class of admissible functions* $\mathcal{F} \subseteq \{f : \mathcal{X}^n \to \mathcal{X}^m\}$ *for some* $n, m = \mathsf{poly}(\lambda)$*.*

$\mathsf{Com}(\mathsf{ck}, \boldsymbol{x}; r) \to (C, \mathsf{aux})$ *on input a vector* $\boldsymbol{x} \in \mathcal{X}^n$ *and (possibly) a randomness* r*, outputs a commitment* C *and related auxiliary information* aux*.* r *can be omitted and randomly sampled.*

$\mathsf{Open}(\mathsf{ck}, \mathsf{aux}, f) \to \pi$ *on input an auxiliary information* aux *and a function* $f \in \mathcal{F}$*, outputs an opening proof* π*.*

$\mathsf{Ver}(\mathsf{ck}, C, f, \boldsymbol{y}, \pi) \to b \in \{0, 1\}$ *on input a commitment* C*, an opening proof* π*, a function* $f \in \mathcal{F}$ *and a value* $\boldsymbol{y} \in \mathcal{X}^m$*, accepts* $(b = 1)$ *or rejects* $(b = 0)$*.*

Correctness. FC *is correct if for any* $n, m \in \mathbb{N}$*, all* $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, n)$*, any* $f : \mathcal{X}^n \to \mathcal{X}^m$ *in the class* $\mathcal{F}$*, and any vector* $\boldsymbol{x} \in \mathcal{X}^n$*, if* $(C, \mathsf{aux}) \leftarrow \mathsf{Com}(\mathsf{ck}, \boldsymbol{x})$*, then it holds* $\mathsf{Ver}(\mathsf{ck}, C, f, f(\boldsymbol{x}), \mathsf{Open}(\mathsf{ck}, \mathsf{aux}, f)) = 1$ *with probability* 1*.*

Succinctness/Compactness. *A functional commitment* FC *is* succinct *if there exists a fixed polynomial* $p(\lambda) = \mathsf{poly}(\lambda)$ *such that for any* $n, m = \mathsf{poly}(\lambda)$*, any admissible function* $f \in \mathcal{F}$ *such that* $f : \mathcal{X}^n \to \mathcal{X}^m$*, honestly generated commitment key* $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, n, m)$*, vector* $\boldsymbol{x} \in \mathcal{X}^n$*, commitment* $(C, \mathsf{aux}) \in \mathsf{Com}(\mathsf{ck})$ *and opening* $\pi \leftarrow \mathsf{Open}(\mathsf{ck}, \mathsf{aux}, f)$*, it holds that* $|C| \leq p(\lambda)$ *and* $|\pi| \leq p(\lambda) \cdot m$*. Furthermore, we say that* FC *is* compact *if* $|\pi| \leq p(\lambda)$*.*

The considered FCs used in the pairing based construction of Section 3 also satisfy a nice property which allows to additively combine commitments.

Additive-homomorphic FC [20]. Let FC be a functional commitment scheme where $\mathcal{X}$ is a ring. Then FC is *additive homomorphic* if there exist deterministic algorithms $\mathsf{FC.Add}(\mathsf{ck}, C_1, \ldots, C_n) \to C$, $\mathsf{FC.Add}_{\mathsf{aux}}(\mathsf{ck}, \mathsf{aux}_1, \ldots, \mathsf{aux}_n) \to \mathsf{aux}$ and $\mathsf{FC.Add}_r(\mathsf{ck}, r_1, \ldots, r_n) \to r$ such that for any $\boldsymbol{x}_i \in \mathcal{X}$ and $(C_i, \mathsf{aux}_i) \leftarrow \mathsf{Com}(\mathsf{ck}, \boldsymbol{x}_i; r_i)$, if $C \leftarrow \mathsf{FC.Add}(\mathsf{ck}, C_1, \ldots, C_n)$, $\mathsf{aux} \leftarrow \mathsf{FC.Add}_{\mathsf{aux}}(\mathsf{ck}, \mathsf{aux}_1, \ldots, \mathsf{aux}_n)$, and $r \leftarrow \mathsf{FC.Add}_r(\mathsf{ck}, r_1, \ldots, r_n)$, then $(C, \mathsf{aux}) = \mathsf{Com}(\mathsf{ck}, \sum_{i=1}^n \boldsymbol{x}_i; r)$.

Security. The common security notion for FCs is the *evaluation binding*, which consists in the hardness of computing two different opening proofs for two different values which pass the verification step for the same function. As proven in [20], for the security of construction of the FC-based Homomorphic Signature in Section 3, it is enough that the FC satisfies the weaker notion of security below.

Definition 2 (Weak Evaluation Binding [20]). *For any PPT adversary* $\mathcal{A}$

$$\mathbf{Adv}^{\mathsf{wEvBind}}_{\mathcal{A},\mathsf{FC}}(\lambda) := \Pr\left[\begin{matrix}(C,\cdot) = \mathsf{Com}(\mathsf{ck},\boldsymbol{x};r) \\ \wedge\ y \neq f(\boldsymbol{x}) \wedge \\ \mathsf{Ver}(\mathsf{ck},C,f,\boldsymbol{y},\pi) = 1\end{matrix} : \begin{matrix}\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, n) \\ (\boldsymbol{x},r,f,\boldsymbol{y},\pi) \leftarrow \mathcal{A}(\mathsf{ck})\end{matrix}\right] = \mathsf{negl}(\lambda)$$

2.2 Lattices

Lattices. A $m-$dimensional lattice $\mathcal{L}$ of rank n is defined as the set of linear combinations of a set of vectors $\mathbf{B} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n], \boldsymbol{v}_i \in \mathbb{R}^m$, with coefficients in $\mathbb{Z}$, i.e. $\mathcal{L} = \{\mathbf{B}\boldsymbol{z} : \boldsymbol{z} \in \mathbb{Z}^m\}$. $\mathbf{B}$ is called a basis for $\mathcal{L}$.

Usually, we use the term *short basis* to refer to a matrix with norm smaller than a certain bound.

Definition 3 (SIS problem). *Given parameters* n, m, q, β *and a matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$*, the Short Integer Solution problem* (SIS) *consists in finding a non-zero vector* $\boldsymbol{u} \in \mathbb{Z}^m$ *such that* $\mathbf{A}\boldsymbol{u} = \mathbf{0} \mod q$ *and* $||\boldsymbol{u}|| \leq \beta$.[7] *The* SIS *problem is said to be hard if the probability of finding such a* $\boldsymbol{u}$ *is negligible, for all polynomially bounded adversaries.*

2.3 Homomorphic Signatures

In this work we compare several Homomorphic Signature schemes (HS). Informally, an HS allows a user to sign a dataset $(x_1, \ldots, x_n)$ and obtain a set of signatures $(\sigma_1, \ldots, \sigma_n)$ for which an untrusted server can compute a function $y = f(x_1, \ldots, x_n)$ and a signature σ. If correctly computed, (σ, f, y) can be correctly verified with the user verification key certifying that $y = f(\boldsymbol{x})$ (without the need to know $\boldsymbol{x}$). In this subsection we recall the definition of a Multi-Data Homomorphic Signature scheme and its most basic security notion from [30]. The definition considers more datasets. In what follows Δ denotes the datased identifier and $\mathcal{X}$ the message space.

Definition 4 (Multi-data Homomorphic Signature Scheme [30]). *A Multi–data homomorphic signature scheme consists of poly-time algorithms* $(\mathsf{PrmsGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Ver})$ *with the following syntax.*

$\mathsf{PrmsGen}(1^\lambda, \mathsf{data}) \to \mathsf{prms}$ *on input the security parameter* λ *and some data* data *specific of the construction, generates public parameters* prms.

$\mathsf{KeyGen}(1^\lambda, \mathsf{prms}) \to (\mathsf{pk}, \mathsf{sk})$ *on input the security parameter* λ *and* prms*, returns a public verification key* pk *and a secret signing key* sk.

$\mathsf{Sign}(\mathsf{sk}, \Delta, (x_1, \ldots, x_N)) \to (\sigma_1, \ldots, \sigma_N)$ *on input* $\bar{\boldsymbol{x}} \in \mathcal{X}^*$ *under a dataset identifier* $\Delta \in \{0,1\}^*$*, returns a tuple of signature* $(\sigma_1, \ldots, \sigma_N)$.

[7] Typically chosen norms are ℓ_2 and ℓ_∞. GVW15 opts for ℓ_∞.

$\mathsf{Eval}(\mathsf{pk}, g, \Delta, ((x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell))) \to \sigma^*$ *takes a set of message/signature pairs under a dataset identifier* $\Delta \in \{0,1\}^*$ *and homomorphically computes a signature* σ^*.[8]

$\mathsf{Ver}(\mathsf{pk}, g, y, \Delta, \sigma^*) \to b \in \{0,1\}$ *verifies that* $y \in \mathcal{X}$ *is indeed the output of* g *over the data signed with the dataset identifier* Δ.

Correctness of Signing. *For any* $\mathsf{prms} \in \mathsf{PrmsGen}(1^\lambda, \mathsf{data})$, $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$, *any* $(x_1, \ldots, x_N) \in \mathcal{X}^N$, *any* $\Delta \in \{0,1\}^*$ *and any* $(\sigma_1, \ldots, \sigma_N) \in \mathsf{Sign}(\mathsf{sk}, \Delta, (x_1, \ldots, x_N))$, *then* $\mathsf{Ver}(\mathsf{pk}, \mathsf{id}_i, x_i, \Delta, \sigma_i) = 1$, *where* id_i *is the identity function which sends* $(x_1, \ldots, x_N)$ *to* x_i.

Correctness of Evaluation. *For any circuits* $h_1, \ldots, h_\ell$ *with* $h_i : \mathcal{X}^N \to \mathcal{X}$ *and any circuit* $g : \mathcal{X}^\ell \to \mathcal{X}$, *any* $(x_1, \ldots, x_\ell) \in \mathcal{X}^\ell$, *any* $\Delta \in \{0,1\}^*$ *and any* $(\sigma_1, \ldots, \sigma_\ell)$*:*

$$\left\{ \begin{array}{c} \{\mathsf{Ver}(\mathsf{pk}, h_i, x_i, \Delta, \sigma_i) = 1\}_{i \in [\ell]} \\ \sigma^* := \mathsf{Eval}(\mathsf{pk}, g, \Delta, (x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell)) \end{array} \right\} \Rightarrow$$
$$\Rightarrow \mathsf{Ver}(\mathsf{pk}, (g \circ \bar{h}), g(x_1, \ldots, x_\ell), \Delta, \sigma^*) = 1.$$

An additional property that HS schemes can satisfy is *succinctness*. Informally, an HS is *succinct* if the Eval output size is smaller than $\mathsf{poly}(\lambda) \cdot \log n$, where n is the input size.

Security. As discussed in the Introduction, for HS schemes we cannot use the basic unforgeability notions for digital signature schemes. In the context of HS, we require that the types of forgery are restricted to signatures that come from the legitimate signer or from the evaluation on them. An analysis and a formal description of the security notions for HS scheme is given in [19]. There, the authors introduce three security notions: *semi-adaptive*, *adaptive* and *strong adaptive* security. The scheme from [30] described in Section 4 guarantees semi-adaptive security[9], while the FC-based schemes recalled in Section 3 satisfy strong adaptive security. For simplicity, we recall here the simpler definition given in [30], which is essentially the semi-adaptive security notion from [19]. We refer to [19] for a more detailed description.

Remark 1. We remark here that in [19] security is defined with respect to (multi)-labeled programs. Since the focus of this paper is on implementations of existing, already proved secure, schemes for the sake of simplicity we rephrase our definition without relying on the somewhat more cumbersome machinery of labeled programs.

[8] g must be an admissible function where the notion of admissible function depends on the specific scheme.

[9] Actually, the base scheme from [30] satisfies only selective security, but in the same work the authors present a construction to extend the scheme in order to achieve a security notion which is essentially equivalent to semi-adaptive security.

Security Experiment [30]. Let HS be as in Definition 4. Consider the following experiment with a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

1. $\mathcal{C}$ runs $\mathsf{prms} \leftarrow \mathsf{PrmsGen}(1^\lambda, \mathsf{data})$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$. pk and prms are then given to $\mathcal{A}$.
2. $\mathcal{A}$ sends an arbitrary number of queries. For each query j
   - $\mathcal{A}$ chooses a fresh and never queried identifier $\Delta_j \in \{0,1\}^*$ and a message vector $(x_{j,1} \dots x_{j,N_j}) \in \mathcal{X}^*$.
   - $\mathcal{C}$ replies to $\mathcal{A}$ with $(\sigma_{j,1}, \dots, \sigma_{j,N_j}) \leftarrow \mathsf{Sign}(\mathsf{sk}, \Delta_j, (x_{j,1} \dots x_{j,N_j}))$.
3. $\mathcal{A}$ chooses a function $g : \mathcal{X}^{N'} \to \mathcal{X}$ and Δ, y', σ'. Let $y := g(x_1, \dots, x_n)$. $\mathcal{A}$ wins the game if:
   - $\mathsf{Ver}(\mathsf{pk}, g, y', \sigma') = 1$;
   - One of the following types of forgeries happens:
     - Type-1: $\Delta \neq \Delta_j$ for any j or $\Delta = \Delta_j$ for some j and $N' \neq N_j$
     - Type-2: $\Delta = \Delta_j$ for some j such that g is admissible on $x_{j,1}, \dots, x_{j,N'}$ and $y' \neq g(x_{j,1}, \dots, x_{j,N'})$

HS is said to be *secure* if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins is $\mathsf{negl}(\lambda)$ in the security experiment above.

3 Pairing based Homomorphic Signatures

The aim of this section is to introduce the two pairing based schemes of CFT22 and BCFL23 that build upon additive-homomorphic FCs. The main idea is that the signer generates a commitment $C_{\boldsymbol{x}}$ to his dataset and a digital signature $\sigma_{C_{\boldsymbol{x}}}$ on the commitment and gives it to the server. The server can then compute a function f by giving to the verifier the pair $(C_{\boldsymbol{x}}, \sigma_{C_{\boldsymbol{x}}})$ and an opening of $C_{\boldsymbol{x}}$ to the function f. Then the verifier can use the signer public key to verify the server computation. In addition, to dispense the signer from having to commit to the dataset all at once, [20] proposes the use of additive-homomorphic FC and a linearly homomorphic signature (LHS) in the commitment space. Concretely the LHS used in [20] is the structure preserving linearly homomorphic scheme from [23]. The LHS supports $\mathsf{FC.Add}$ for evaluations. For what concerns the security, as proven in [20], the HS is strongly-adaptive secure if the underlying LHS is strongly-adaptive secure and the FC satisfies weak evaluation binding. We implement the LHS construction presented in the original paper, which supports commitments to group elements for the pairing based FC under consideration. The LHS implemented is built from any (standard) digital signature scheme. The algorithms of the resulting HS construction are summarized in Figure 1.

Short overview of the implemented FC The first FC scheme (CFT22) we use to instantiate an HS is the one presented in the Section 4 of [20]. This scheme supports a commitment opening to multiple (multivariate) polynomials of bounded constant degree (fixed at setup time) with a single *compact* proof and its security is proven under the standard Diffie-Hellman exponent assumption. On the other hand, the second FC scheme (BCFL23), proposed in [7], supports

$\mathsf{KeyGen}(1^\lambda, [n])$

$\mathsf{ck} \leftarrow \mathsf{FC.Setup}(1^\lambda, n, m)$

$(\mathsf{sk}_{\mathsf{LHS}}, \mathsf{pk}_{\mathsf{LHS}}) \leftarrow \mathsf{LHS.KeyGen}(1^\lambda, [n])$

$\mathsf{pk} := (\mathsf{pk}_{\mathsf{LHS}}, \mathsf{ck}), \mathsf{sk} := \mathsf{sk}_{\mathsf{LHS}}$

return $(\mathsf{sk}, \mathsf{pk})$

$\mathsf{Sign}(\mathsf{sk}, \Delta, i, x_i)$

Let $\boldsymbol{e}_i$ s.t. $e_{i,i} = 1, e_{i,j} = 0\ \forall i \neq j$

$(C_i, \mathsf{aux}_i) \leftarrow \mathsf{FC.Com}(\mathsf{ck}, x_i \cdot \boldsymbol{e}_i)$

$\hat{\sigma}_i \leftarrow \mathsf{LHS.Sign}(\mathsf{sk}_{\mathsf{LHS}}, \Delta, i, C_i)$

return $\sigma_i := (\hat{\sigma}_i, C_i, \mathsf{aux}_i, i)$

$\mathsf{Eval}(\mathsf{pk}, f, \sigma_1, \ldots, \sigma_t)$

$C \leftarrow \mathsf{FC.Add}(\mathsf{ck}, C_1, \ldots, C_t)$

$\mathsf{aux} \leftarrow \mathsf{FC.Add}_{\mathsf{aux}}(\mathsf{ck}, \mathsf{aux}_1, \ldots, \mathsf{aux}_t)$

$\hat{\sigma} \leftarrow \mathsf{LHS.Eval}(\mathsf{pk}_{\mathsf{LHS}}, \mathsf{FC.Add}, \hat{\sigma}_1, \ldots, \hat{\sigma}_t)$

$\pi \leftarrow \mathsf{FC.Open}(\mathsf{ck}, \mathsf{aux}, \hat{f}_{\boldsymbol{i}})$

return $\sigma_{f,y} := (\hat{\sigma}, C, \pi_f)$

$\mathsf{Ver}(\mathsf{pk}, (f, \boldsymbol{i}), \Delta, \boldsymbol{y}, \sigma)$

$b_{\mathsf{LHS}} \leftarrow \mathsf{LHS.Ver}(\mathsf{pk}_{\mathsf{LHS}}, (\mathsf{FC.Add}, \boldsymbol{i}), \Delta, C, \hat{\sigma})$

if $\sigma = (\hat{\sigma}, C, \mathsf{aux}, i)$

$\pi \leftarrow \mathsf{FC.Open}(\mathsf{ck}, \mathsf{aux}, \hat{f}_{id,i})$

$b_{\mathsf{FC}} \leftarrow \mathsf{FC.Ver}(\mathsf{ck}, C, \hat{f}_{\boldsymbol{i}}, \boldsymbol{y}, \pi)$

return $b_{\mathsf{FC}} \wedge b_{\mathsf{LHS}}$

Fig. 1: HS from additive FC and LHS for FC.Add.

the evaluation of arbitrary arithmetic circuits with a maximal *width*, the opening proof is *succinct* and its security is proven under a new falsifiable assumption called n-HiKer. Both schemes are additive-homomorphic and have amortized efficient verification.

4 A Lattice based Homomorphic Signature

In this section we recall the Lattice based Homomorphic Signature of GVW15. The authors present two schemes, one with larger public parameters in the standard model and one with shorter public parameters in the random oracle model. The construction of both HS schemes relies upon a primitive called *Homomorphic Trapdoor Function* (HTDF), and their security is guaranteed under the SIS assumption. In the HS only the depth d of the circuits is fixed in setup, and the protocol can evaluate arbitrary circuits. The HS allows full composability, meaning that outputs of some evaluations can be used as inputs to a new evaluation which can be verified using the verification key of the legitimate signer.

4.1 Homomorphic Trapdoor Function HTDF

The notion of Homomorphic Trapdoor Function (HTDF) is introduced in GVW15. We recall the formal definition and refer to the original paper for a more detailed description[10].

Definition 5 (Homomorphic Trapdoor Function [30]). *A* HTDF *is a tuple of five algorithms* $(\mathsf{HTDF.KeyGen}, f, \mathsf{Inv}, \mathsf{HTDF.Eval}^{in}, \mathsf{HTDF.Eval}^{out})$ *where:*

[10] The original paper also presents an HTDF construction, which is the one we used in our implementation.

- $\mathsf{HTDF.KeyGen}$ *on input the security parameter* λ *returns a public key* pk *and a secret key* sk. λ *also defines the index space* $\mathcal{X}$*, the input space* $\mathcal{U}$*, the output space* $\mathcal{V}$ *and an input distribution* $D_{\mathcal{U}}$ *over* $\mathcal{U}$*. It is required that membership tests in* $\mathcal{X}$,$\mathcal{U}$,$\mathcal{V}$ *and uniform sampling in* $\mathcal{V}$ *can be done efficiently.*
- $f_{\mathsf{pk},x} : \mathcal{U} \to \mathcal{V}$ *a deterministic function indexed by* $x \in \mathcal{X}$ *and* pk*, on input an element in* $\mathcal{U}$ *returns an element in* $\mathcal{V}$*.*
- $\mathsf{Inv}_{\mathsf{sk},x} : \mathcal{V} \to \mathcal{U}$ *a probabilistic inversion function indexed by* $x \in \mathcal{X}$ *and* sk*, on input an element in* $\mathcal{V}$ *returns a preimage in* $\mathcal{U}$*.*
- $\mathsf{HTDF.Eval}^{in}$ *a deterministic homomorphic evaluation algorithm which, on input* $\{(x_i, u_i)\}_{i\in[\ell]}$*, where* $u_i \in \mathcal{U}$*,* $x_i \in \mathcal{X}$*, for each* $i \in [\ell]$*, and a circuit* $g : \mathcal{X}^\ell \to \mathcal{X}$*,* $\mathsf{HTDF.Eval}^{in}$*, returns* $u^* := \mathsf{HTDF.Eval}^{in}(g, (x_1, u_1), \ldots, (x_\ell, u_\ell))$ *where* $u^* \in \mathcal{U}$*.*
- $\mathsf{HTDF.Eval}^{out}$ *a deterministic homomorphic evaluation algorithm which, on input* $\{v_i\}_{i\in[\ell]}$*, where* $v_i \in \mathcal{V}$*, for each* $i \in [\ell]$*, and a circuit* $g : \mathcal{X}^\ell \to \mathcal{X}$*,* $\mathsf{HTDF.Eval}^{out}$ *, returns* $v^* := \mathsf{HTDF.Eval}^{out}(g, v_1, \ldots, v_\ell)$*, where* $v^* \in \mathcal{V}$*.*

Correctness. *Let* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*, let* $g : \mathcal{X}^\ell \to \mathcal{X}$ *a circuit,* $x_1, \ldots, x_\ell \in \mathcal{X}$ *and* $y := g(x_1, \ldots, x_\ell)$*. Let* $u_1, \ldots, u_\ell \in \mathcal{U}$ *and define* $v_i := f_{\mathsf{pk},x_i}(u_i)$ *for each* $i \in [\ell]$*. Let* $u^* := \mathsf{HTDF.Eval}^{in}(g, (x_1, u_1), \ldots, (x_\ell, u_\ell))$ *and* $v^* := \mathsf{HTDF.Eval}^{out}(g, v_1, \ldots, v_\ell)$*, then correctness of homomorphic evaluations requires that* $u^* \in \mathcal{U}$ *and* $f_{\mathsf{pk},y}(u^*) = v^*$.[11]

An HTDF can be seen as a homomorphic equivocable commitment scheme, where knowing the secret key it is possible to open any commitment v to u for any index x. This motivates the security notion below for a HTDF, which is similar to the binding property of a commitment scheme.

Definition 6 (HTDF Security [30]). *An* HTDF *is said to be secure if for any PPT adversary* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *finds two inputs* $u, u' \in \mathcal{U}$ *and* $x \neq x' \in \mathcal{X}$ *such that* $f_{\mathsf{pk},x}(u) = f_{\mathsf{pk},x'}(u')$ *is* $\mathsf{negl}(\lambda)$*.*

4.2 Lattice based HS scheme from HTDF

[30] presents a way to use an HTDF as a building block to construct an Homomorphic Signature scheme. A concrete HTDF can be instantiated using existing algorithms such the ones in [37], which is the construction we chose for our implementation of the HS from GVW15 discussed in Subsection 5.2. More precisely, the implemented scheme, which is built for a single dataset, satisfies a notion of security called *selective security* where an adversary chooses its messages $(x_1, \ldots, x_\ell)$ before knowing the public parameters and the public key. [30] also present a construction to transform the basic scheme to a more complex one which satisfies full security, which is essentially the semi-adaptive security.

[11] The evaluated function g must be an admissible function, otherwise the correctness property is not guaranteed. See description in Section 4.2 for more details.

We implement the single-data scheme in the standard model and we discuss our implementation in Subsection 5.2.

Homomorphic Signature construction. The GVW15 single-data HS scheme inherits some characteristic of the underlying HTDF. More in details, we give a short description of noise analysis to introduce these inherited properties and to explain what is an admissible function in GVW15. Any function used in the evaluation algorithm is expressed as an arithmetic circuit. The HS signatures are matrices in $\mathcal{U}$, as given by the construction of the HTDF, with an associated noise level β equal to their ℓ_∞ norm. Any evaluation outputs a signature with a noise-level which depends on the evaluated function and the messages[12]. After some evaluation the output has some noise β. The HS is parametrized by a threshold noise level β_{max} which must not be exceeded by the noise level β of an evaluated signature, otherwise the correctness of the HTDF and then of the HS is not guaranteed. The HS of GVW15 inherits from HTDF both β_{max} and the set of admissible functions. The correctness of the homomorphic evaluations in the HS comes from the corresponding ones in the HTDF.

GVW15 scheme. The single-data HS construction from GVW15 that we implement is described below:
Let $\mathsf{HTDF} = (\mathsf{HTDF.KeyGen}, f, \mathsf{Inv}, \mathsf{HTDF.Eval}^{in}, \mathsf{HTDF.Eval}^{out})$ be an HTDF with index space $\mathcal{X}$, input space $\mathcal{U}$, output space $\mathcal{V}$ and an input distribution $D_{\mathcal{U}}$. The GVW15 HS is a tuple of algorithms $\mathsf{HS} = (\mathsf{PrmsGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{Eval})$ where Ver is divided in two subprocedures $(\mathsf{Ver}^*, \mathsf{Process})$. HS has message space $\mathcal{X}$ and:

- $\mathsf{PrmsGen}(1^\lambda, 1^N) \to \mathsf{prms}$: return $\mathsf{prms} = (v_1, \ldots, v_N) \xleftarrow{\$} \mathcal{V}^N$.
- $\mathsf{KeyGen}(1^\lambda, \mathsf{prms}) \to (\mathsf{pk}, \mathsf{sk})$: set $(\mathsf{pk}, \mathsf{sk}) = (\mathsf{pk}', (\mathsf{prms}, \mathsf{sk}'))$ where $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{HTDF.KeyGen}(1^\lambda)$
- $\mathsf{Sign}(\mathsf{sk}, x_1, \ldots, x_N) \to (\sigma_1, \ldots, \sigma_N)$: sample $u_i \leftarrow \mathsf{Inv}_{\mathsf{sk}', x_i}(v_i)$ and set $\sigma_i := u_i$ for $i \in [N]$.
- $\mathsf{Eval}(\mathsf{pk}, g, (x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell)) \to \sigma^*$: run $\mathsf{HTDF.Eval}^{in}(\mathsf{pk}', g, (x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell))$
- $\mathsf{Ver}(\mathsf{pk}, g, y, \sigma^*) \to \{0, 1\}$:[13]
  - $\alpha_g \leftarrow \mathsf{Process}(\mathsf{prms}, g)$: set $\alpha_g \leftarrow \mathsf{HTDF.Eval}^{out}(\mathsf{pk}', g, v_1, \ldots, v_N)$.
  - $\mathsf{Ver}^*(\mathsf{pk}, \alpha_g, y, \sigma^*)$: verify $f_{pk', y}(\sigma) \stackrel{?}{=} \alpha_g$. If true accept, else reject.

Security. We defined the full security of a multi-data scheme in Section 2, which corresponds to the semi-adaptive security notion. For the single-data protocol the security experiment takes into account only one dataset and a possible forgery is only of the second type, i.e. when datasets are all the same one. Anyway,

[12] The noise level grows after any evaluation gate of the circuit.

[13] In the original work [30], the names of Ver and Ver^* are switched and the construction is presented with the procedures Process and Ver^* taken separately and not unified as in Ver here. To be consistent with the definition we gave in Section 2, we opted to present the construction with a unified procedure where Ver^* has the role of Ver in the original paper.

the basic protocol we considered satisfies a weaker notion of security where the adversary chooses messages to ask for signatures before seeing the public key and the parameters.

5 Implementations

As our main contribution in this work, we implemented the three constructions of Sections 3 and 4 in order to compare them in terms of execution time and bandwidth consumption. Additionally, we carried on a feasibility study for some commonly used statistical functions to understand their applicability in real-world scenarios. Our implementations were written in Rust programming language, along with several additional libraries we will cite later[14].

5.1 Pairing based schemes

As previously mentioned, CFT23 and BCFL23 are based on pairings. For the implementation of both schemes, we used the open-source *blstrs* cryptographic library provided by Protocol Labs [39]. This is a Rust binding for the well-known *blst* library [40], written in C and Assembly, which provides an efficient implementation of the Barreto-Lynn-Scott pairing-friendly elliptic curve [9] with embedding degree 12 and a 381 bits modulus (BLS12-381). The BLS12-381 curve was instantiated from the BLS12 curve family by appropriately choosing the instantiation parameter to minimise its Hamming weight, which is useful for achieving the best performance on the Miller's Weil pairing algorithm (also known as Miller's Loop), while maximizing the size of the field modulus q. This allows fast 32/64-bit arithmetic. BLS12-381 was originally designed to provide a 128 bit security level, however it is known that the extended tower number field sieve (exTNFS) [32] reduces its security to approximately 100 bits [8]. For each of the aforementioned schemes a library was developed using a modular approach. Two different modules have been created: one for the FC scheme and one for the LHS construction. These modules were then called up in another module specifically created for the HS construction. For both schemes we implemented the LHS proposed in [20] using BLS [13] as standard digital signature, namely the implementation of the *blst* [40] library.

5.2 Lattice based scheme

We implemented the lattice based scheme (GVW15) using the qFALL library [38], which provides mathematical primitives for lattice based cryptography and the implementation of the trapdoor functions proposed in [37]. qFALL uses the optimized widespread FLINT library [41] as back-end for the arithmetic on standard rings. First of all, we tried to find proper parameters, using the Lattice Estimator project [2], to instantiate the scheme in order to provide a security of

[14] The source code is available upon request of the reviewers.

approximately 100 bits. After the early experiments we realised that with such parameters, the execution time of the benchmarks would be totally unacceptable, mainly due to the KeyGen and Sign algorithms. Specifically, in addition to compute a parity matrix $\mathbf{A}$ and an associated trapdoor td, the KeyGen algorithm also generates a short base of the lattice of interest using td and precomputes its Gram-Schmidt Orthogonalisation. This is necessary to optimise the execution of the SamPre algorithm during the signing phase. Unfortunately, it is the Gram-Schmidt Orthogonalisation that is the most time-consuming operation in our KeyGen implementation. As previously outlined, the Sign algorithm simply invokes the Inv algorithm, whose execution time is exclusively dependent on the parameters selected for the SIS instance. In fact, this algorithm solves a certain number of systems of linear equations depending on the chosen SIS parameters. Using the qFALL library, our implementation of this algorithm employs the Gaussian elimination method. Despite this optimization, the Sign algorithm remains very time-consuming due to the size of the involved matrices: more precisely if n and q are the parameters for the SIS instance, then we have to solve $n\lceil \log_2 q \rceil$ systems of linear equations modulus q. For these reasons, we decided to use lower non-secure parameters, namely $n = 16$ and $q = 2601023$. According to the Lattice Estimator, this would give a security of about 30 bits. Nevertheless, as we shall see in Section 6, the scheme performs poorly. The scheme was implemented in a generic way, enabling the evaluation of both boolean and arithmetic circuits as outlined in the original article. However, our benchmarks were carried out on arithmetic circuits to reduce execution time and bandwidth consumption. Indeed, using boolean circuits would have posed challenges in expressing the chosen functions and produced more signatures (one for each bit of the vector elements), leading to wasted bandwidth and CPU cycles.

5.3 On the possibility of a parallel implementation

For the sake of simplicity, the libraries were implemented using a single-threaded approach. While a parallel implementation could potentially enhance the efficiency of many operations in the considered constructions, we believe it would not fundamentally subvert the final conclusions. More specifically, there are two levels of parallelism that we can consider: one for the basic mathematical operations (e.g. multi-precision integer arithmetic, elliptic curve arithmetic, pairing computation, polynomial and matrix operations etc.) and one that is inherent to the protocol that we implemented. Regarding the first level, as mentioned before, standard libraries have been employed for the basic mathematical operations but, unfortunately, these do not provide any parallelism support. On the other hand, for the second level of parallelism, we conducted an analysis of the implemented algorithms in order to evaluate the impact of a parallel approach. With regard to the pairing based HS, all algorithms are amenable to parallelization: this would result in a reduction of the execution time by an expected factor equal to the number of used threads[15]. We would have the same reduction

[15] This assertion can be corroborated by inspecting the FC constructions [20, 7].

factor in the case of GVW15, especially in the KeyGen algorithm, using a parallel Gram-Schmidt Orthogonalization implementation (we used the FLINT one), and in the Sign algorithm by solving the systems of linear equations in parallel. Obviously, even in the case of the Eval algorithm, we can have the same gain by parallelizing the evaluation of the gates when it is possible. If we keep out the preprocessing phase for the amortized verification, the Ver algorithm is the least likely to benefit from this level of parallelism.

5.4 Benchmarks

First of all, we implemented a benchmark unit to compare the schemes in terms of execution time. We used the Divan library [42] which provides a good approximation of the strategy proposed in [24] for benchmarking even in noisy environments. We also considered the use of the Criterion library: indeed it is considered the *de facto* standard for Rust benchmarks. A wider flexibility in the test management but also a better accuracy guaranteed by the native use of the Time Stamp Counter (TSC), determined our final choice. For each benchmark, we collected as many samples as possible for at least 60 seconds. This guaranteed a proper number of runs on fine-grain operations but also a dynamic experimental environment on heterogeneous operations. In experiments where we collected more than one sample, the value in Section 6 represents the median of the collected samples. Our benchmarks keep track of the bandwidth usage among the parties in the implemented schemes. Due to the complexity of the involved data structures, we decided to use as metric the size of the serialization obtained by means of the Concise Binary Object Representation (CBOR)[14] provided by the Serde serialization library. Benchmark units have been implemented for two representative functions in statistics: the weighted sum and the variance. The first one gives us an understanding of the behaviour of constructions in the linear case, the second one in the quadratic case[16].

6 Experimental results

In this section we report the benchmark results we collected in our experiments. In all the executions we evaluated the chosen statistic functions on the same randomly-generated vector of 8-bit unsigned integers[17]. All our benchmarks were executed on a server equipped with an Intel(R) Xeon(R) Gold 6238R CPU running at 2.20 GHz and 24 GiB of RAM. Figures 2 and 3 show our benchmark results for, respectively, the weighted sum and variance functions. As the collected data are spread over a wide range, we chose to use a logarithmic scale in our graphs. They give an idea of the running times of the main operations and of

[16] We implemented the variance using a pair of grade 2 polynomials omitting the problematic division operation for the public and function-related parameter n.

[17] This choice is mainly related to the toy parameters we were forced to use in GVW15: indeed the other two schemes would be able to manage also 64-bit scalars with the chosen keys.

the sizes of the generated signatures and proofs. The size of the dataset ranges from 5 to 40 elements. Dashed lines were used in the lattice based scheme to indicate the use of very small and insecure parameters. When interpreting these graphs, it is important to consider this aspect, particularly when the GVW15 scheme appears to be comparable to others in terms of evaluation and verification. More specifically, regarding the variance function shown in Figure 3 and focusing solely on pairing based constructions, it can be observed that the key generation of CFT22 is significantly faster compared to BCFL23. Additionally, the evaluation and verification functionalities of CFT22 are quicker than BCFL23 by an average factor of 12 and 23 times, respectively. On the other hand, BCFL23 is three times more efficient in the signing phase and produces signatures that are three times smaller, albeit with slightly more concise proofs.

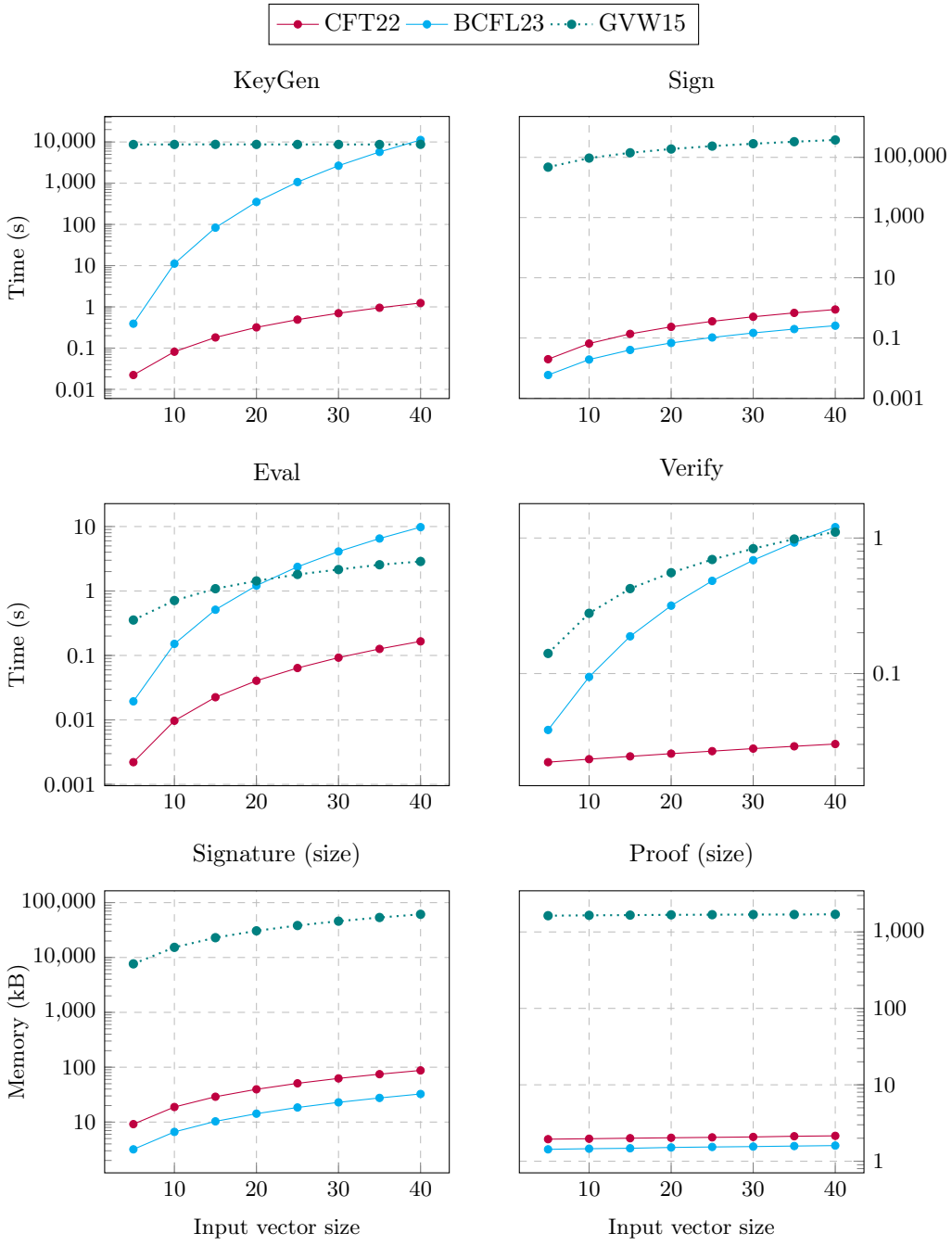

Fig. 2: Weighted Sum Function

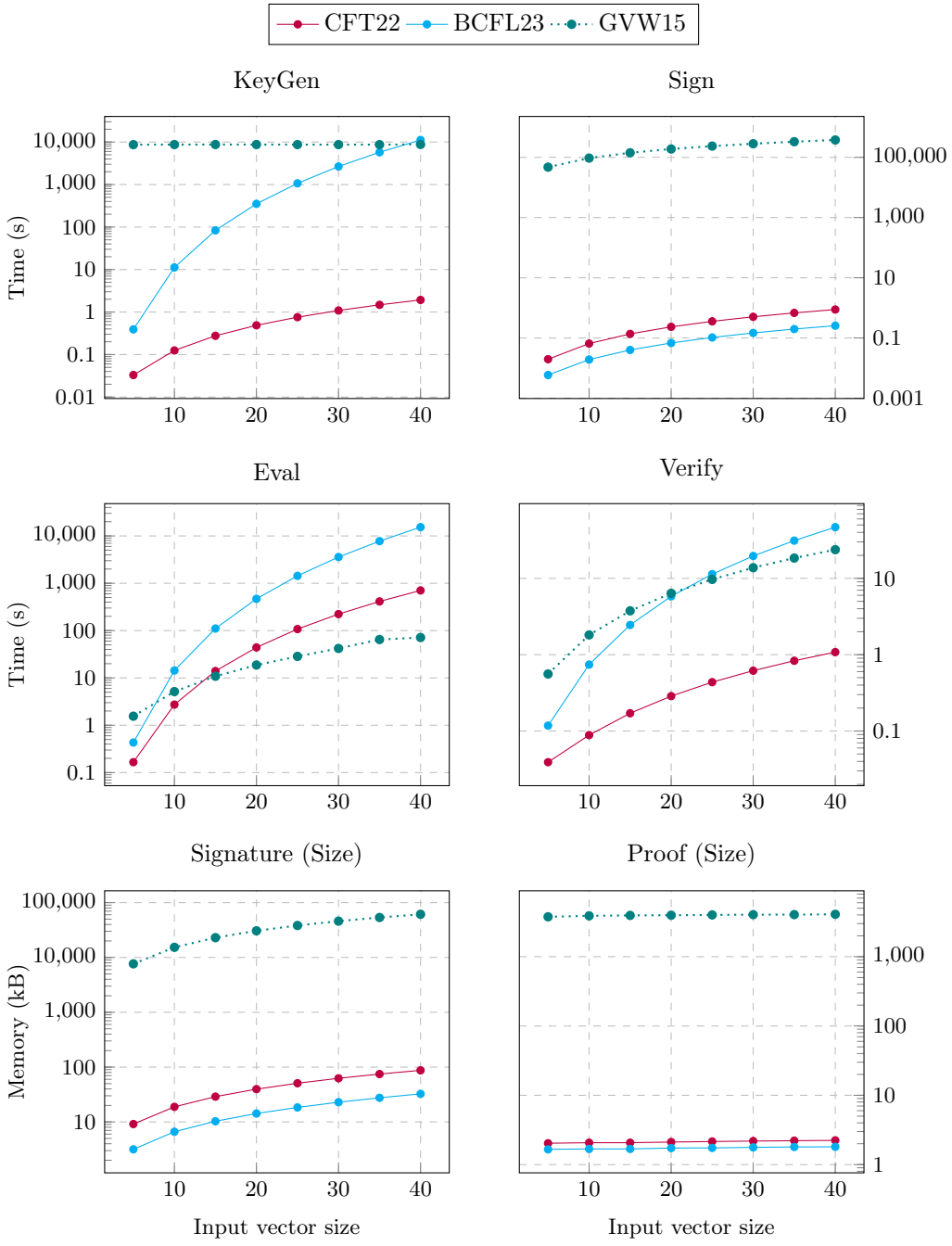

Fig. 3: Variance Function

References

1. Agrawal, S., Boneh, D.: Homomorphic MACs: MAC-based integrity for network coding. In: Abdalla, M., Pointcheval, D., Fouque, P.A., Vergnaud, D. (eds.) ACNS 09. LNCS, vol. 5536, pp. 292–305. Springer, Heidelberg (Jun 2009). https://doi.org/10.1007/978-3-642-01957-9_18
2. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**(3), 169–203 (Oct 2015). https://doi.org/10.1515/jmc-2015-0016, http://dx.doi.org/10.1515/jmc-2015-0016
3. Attrapadung, N., Libert, B.: Homomorphic network coding signatures in the standard model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 17–34. Springer, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19379-8_2
4. Attrapadung, N., Libert, B., Peters, T.: Computing on authenticated data: New privacy definitions and constructions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 367–385. Springer, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34961-4_23
5. Attrapadung, N., Libert, B., Peters, T.: Efficient completely context-hiding quotable and linearly homomorphic signatures. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 386–404. Springer, Heidelberg (Feb / Mar 2013). https://doi.org/10.1007/978-3-642-36362-7_24
6. Backes, M., Fiore, D., Reischuk, R.M.: Verifiable delegation of computation on outsourced data. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 863–874. ACM Press (Nov 2013). https://doi.org/10.1145/2508859.2516681
7. Balbás, D., Catalano, D., Fiore, D., Lai, R.W.F.: Chainable functional commitments for unbounded-depth circuits. In: Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part III. pp. 363–393 (2023). https://doi.org/10.1007/978-3-031-48621-0_13, https://doi.org/10.1007/978-3-031-48621-0_13
8. Barbulescu, R., Duquesne, S.: Updating key size estimations for pairings. Journal of Cryptology **32**(4), 1298–1336 (Oct 2019). https://doi.org/10.1007/s00145-018-9280-5
9. Barreto, P.S.L.M., Lynn, B., Scott, M.: Constructing elliptic curves with prescribed embedding degrees. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 02. LNCS, vol. 2576, pp. 257–267. Springer, Heidelberg (Sep 2003). https://doi.org/10.1007/3-540-36413-7_19
10. Boneh, D., Freeman, D., Katz, J., Waters, B.: Signing a linear subspace: Signature schemes for network coding. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 68–87. Springer, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00468-1_5
11. Boneh, D., Freeman, D.M.: Homomorphic signatures for polynomial functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (May 2011). https://doi.org/10.1007/978-3-642-20465-4_10
12. Boneh, D., Freeman, D.M.: Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 1–16. Springer, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19379-8_1
13. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology **17**(4), 297–319 (Sep 2004). https://doi.org/10.1007/s00145-004-0314-9

14. Bormann, C., Hoffman, P.E.: Concise Binary Object Representation (CBOR). RFC 8949 (Dec 2020). https://doi.org/10.17487/RFC8949, https://www.rfc-editor.org/info/rfc8949
15. Catalano, D., Fiore, D.: Practical homomorphic MACs for arithmetic circuits. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 336–352. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_21
16. Catalano, D., Fiore, D., Gennaro, R., Nizzardo, L.: Generalizing homomorphic MACs for arithmetic circuits. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 538–555. Springer, Heidelberg (Mar 2014). https://doi.org/10.1007/978-3-642-54631-0_31
17. Catalano, D., Fiore, D., Gennaro, R., Vamvourellis, K.: Algebraic (trapdoor) one-way functions and their applications. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 680–699. Springer, Heidelberg (Mar 2013). https://doi.org/10.1007/978-3-642-36594-2_38
18. Catalano, D., Fiore, D., Nizzardo, L.: Programmable hash functions go private: Constructions and applications to (homomorphic) signatures with shorter public keys. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 254–274. Springer, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-662-48000-7_13
19. Catalano, D., Fiore, D., Nizzardo, L.: On the security notions for homomorphic signatures. In: Preneel, B., Vercauteren, F. (eds.) ACNS 18. LNCS, vol. 10892, pp. 183–201. Springer, Heidelberg (Jul 2018). https://doi.org/10.1007/978-3-319-93387-0_10
20. Catalano, D., Fiore, D., Tucker, I.: Additive-homomorphic functional commitments and applications to homomorphic signatures. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part IV. LNCS, vol. 13794, pp. 159–188. Springer, Heidelberg (Dec 2022). https://doi.org/10.1007/978-3-031-22972-5_6
21. Catalano, D., Fiore, D., Warinschi, B.: Adaptive pseudo-free groups and applications. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 207–223. Springer, Heidelberg (May 2011). https://doi.org/10.1007/978-3-642-20465-4_13
22. Catalano, D., Fiore, D., Warinschi, B.: Efficient network coding signatures in the standard model. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 680–696. Springer, Heidelberg (May 2012). https://doi.org/10.1007/978-3-642-30057-8_40
23. Catalano, D., Marcedone, A., Puglisi, O.: Authenticating computation on groups: New homomorphic primitives and applications. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 193–212. Springer, Heidelberg (Dec 2014). https://doi.org/10.1007/978-3-662-45608-8_11
24. Chen, J., Revels, J.: Robust benchmarking in noisy environments (2016)
25. Desmedt, Y.: Computer security by redefining what a computer is. In: NSPW 1993
26. Fiore, D., Tucker, I.: Efficient zero-knowledge proofs on signed data with applications to verifiable computation on data streams. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 1067–1080. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3560630
27. Freeman, D.M.: Improved security for linearly homomorphic signatures: A generic framework. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 697–714. Springer, Heidelberg (May 2012). https://doi.org/10.1007/978-3-642-30057-8_41

28. Gennaro, R., Katz, J., Krawczyk, H., Rabin, T.: Secure network coding over the integers. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 142–160. Springer, Heidelberg (May 2010). https://doi.org/10.1007/978-3-642-13013-7_9
29. Gennaro, R., Wichs, D.: Fully homomorphic message authenticators. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 301–320. Springer, Heidelberg (Dec 2013). https://doi.org/10.1007/978-3-642-42045-0_16
30. Gorbunov, S., Vaikuntanathan, V., Wichs, D.: Leveled fully homomorphic signatures from standard lattices. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th ACM STOC. pp. 469–477. ACM Press (Jun 2015). https://doi.org/10.1145/2746539.2746576
31. Johnson, R., Molnar, D., Song, D.X., Wagner, D.A.: Homomorphic signature schemes. In: Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2271, pp. 244–262. Springer (2002)
32. Kim, T., Barbulescu, R.: Extended tower number field sieve: A new complexity for the medium prime case. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 543–571. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53018-4_20
33. Lai, R.W.F., Malavolta, G.: Subvector commitments with application to succinct arguments. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 530–560. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26948-7_19
34. Libert, B., Peters, T., Joye, M., Yung, M.: Linearly homomorphic structure-preserving signatures and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 289–307. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_17
35. Libert, B., Peters, T., Joye, M., Yung, M.: Non-malleability from malleability: Simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 514–532. Springer, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_29
36. Libert, B., Ramanna, S.C., Yung, M.: Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In: Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y., Sangiorgi, D. (eds.) ICALP 2016. LIPIcs, vol. 55, pp. 30:1–30:14. Schloss Dagstuhl (Jul 2016). https://doi.org/10.4230/LIPIcs.ICALP.2016.30
37. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_41
38. Porzenheim, L., Beckmann, M., Kramer, P., Milewski, P., Moog, S., Schmidt, M., Siemer, N.: qfall-crypto v0.0. Online: https://github.com/qfall/crypto (Mar 2023), university Paderborn, Codes and Cryptography
39. Protocol Labs: blstrs. Online: https://github.com/filecoin-project/blstrs (2020)
40. Supranational: blst. Online: https://github.com/supranational/blst (2020)
41. team, T.F.: FLINT: Fast Library for Number Theory, https://flintlib.org
42. Vazquez, N.: Divan. Online: https://github.com/nvzqz/divan