Leveled Fully Homomorphic Signatures from Standard Lattices

Daniel Wichs*

June 15, 2014

Abstract

In a homomorphic signature scheme, a user Alice signs some large data x using her secret signing key and stores the signed data on a server. The server can then run some computation $y = g(x)$ on the signed data and homomorphically produce a *short* signature $\sigma_{g,y}$. Anybody can verify the signature using Alice's public verification key and become convinced that y is the correct output of the computation g over Alice's data, without needing to have the underlying data itself.

In this work, we construct the first *leveled fully homomorphic signature* schemes that can evaluate arbitrary circuits over signed data, where only the maximal depth d of the circuit needs to be fixed a priori. The size of the evaluated signature grows polynomially in d, but is otherwise independent of the circuit size or the data size. Our solutions are based on the hardness of the *small integer solution* (SIS) problem, which is in turn implied by the worst-case hardness of problems in standard lattices. We get a scheme in the standard model, albeit with large public parameters whose size must exceed the total size of all signed data. In the random-oracle model, we get a scheme with short public parameters. These results offer a significant improvement in capabilities and assumptions over the best prior homomorphic signature scheme due to Boneh and Freeman (Eurocrypt '11).

As a building block of independent interest, we introduce a new notion called *homomorphic trapdoor functions* (HTDF). We show to how construct homomorphic signatures using HTDFs as a black box. We construct HTDFs based on the SIS problem by relying on a recent technique developed by Boneh et al. (Eurocrypt '14) in the context of attribute based encryption.

1 Introduction

Motivated by the prevalence of *cloud computing*, there has been much interest in cryptographic schemes that allow a user Alice to securely outsource her data to an untrusted remote server (e.g., the cloud), while also allowing the server to perform useful computations over this data. The ground-breaking development of *fully homomorphic encryption* (FHE) by Gentry [Gen09] allows Alice to maintain the *privacy* of her data by encrypting it, while allowing the server to homomorphically perform arbitrary computations over the ciphertexts. In this work, we are interested in the dual question of *authenticity.*

Homomorphic Signatures. A *homomorphic signature* scheme allows Alice to store signed data x on a remote server. The server can then perform computations $y = g(x)$ over this data and homomorphically compute a signature $\sigma_{g,y}$, which certifies that y is the correct output of the computation g. The signature $\sigma_{g,y}$ should be short, with length independent of the size of the data x. Alice can verify the tuple $(g, y, \sigma_{g,y})$ using her public verification key and become convinced that y is indeed the correct output of the computation g over her signed data x, without needing to download the entire data back from the server. In particular, this allows Alice to verify computations over her outsourced data with low communication complexity and with minimal interaction consisting of a single message from the server to Alice. Moreover, the *non-interactive* nature of homomorphic signatures and the fact that they provide *public verifiability*

[*] Northeastern University. E-mail: `wichs@ccs.neu.edu`. Research supported by NSF grants 1347350, 1314722.

makes them useful in settings beyond outsourcing. For example, results of computations over signed data can be posted publicly on a third-party website and verified by any visitor to the website without having to interact with the original owner of the data or the party that performed the computation.

1.1 Related Work

Linearly Homomorphic Schemes. Many prior works consider the question of homomorphic message authentication codes (MACs with private verification) and signatures (public verification) for restricted homomorphisms, and almost exclusively for *linear functions*: [ABC$^+$07, SW08, DVW09, AKK09, AB09, BFKW09, GKKR10, BF11a, AL11, BF11b, CFW12, Fre12]. Such MACs/signautres have interesting applications to *network coding* and *proofs of retrievability.*

Homomorphic Signatures Beyond Linear. Boneh and Freeman [BF11a] were the first to consider homomorphic signatures beyond linear functions, and propose a general definition of such signatures. They present a scheme that can evaluate arbitrary *polynomials* over signed data, where the maximal *degree* k of the polynomial is fixed a priori and the size of the evaluated signature grows (polynomially) in k. If we want to translate this to the setting of circuits, then a circuit of depth d can be represented by a polynomial of degree as high as $k = 2^d$, and therefore the signature size can grow exponentially in the depth of the circuit. The construction is based on the hardness of the *Small Integer Solution* (SIS) problem in *ideal lattices* and has a proof of security in the random-oracle model. Boneh and Freeman pose the challenge of constructing signatures with greater levels of homomorphism, and ideally a *fully homomorphic* scheme that can evaluate arbitrary circuits.

Homomorphic MACs Beyond Linear. There has also been progress in constructing *homomorphic message authentication (MACs)* (private verification) for larger classes of homomorphisms. The work of Gennaro and Wichs [GW13] defines and achieves *fully homomorphic MACs* using fully homomorphic encryption. However, the security of the scheme only holds in a setting *without verification queries* and can completely break down if the attacker has access to a verification oracle allowing him to test whether authentication tags are valid. More recent works [CF13, BFR13, CFGN14] show how to get homomorphic MACs that remain secure in the presence of verification queries, but only for restricted homomorphisms. Currently, the best such schemes allow for the evaluation of polynomials of degree k, where the computational effort grows polynomially with k (but the size of the evaluated signature stays fixed). In particular, homomorphic evaluation is only efficient if $k = \mathsf{poly}(\lambda)$ is polynomial in the security parameter λ. Translated to the context of circuits, this allows for the efficient homomorphic evaluation of circuits whose depth $d = O(\log \lambda)$ is *logarithmic* in the security parameter.[1]

Other Types of Homomorphic Authentication. We also mention works on specific types of homomorphic properties such as *redactable signatures* (see e.g., [JMSW02, ABC$^+$12]) where, given a signature on a long message x, it should be possible to derive a signature on a subset/substring x' of x. The work of [ABC$^+$12] proposes a very general notion of P-homomorphic signature schemes for various predicates P, but efficient constructions are only given for a few specific instances.

Homomorphic Signatures via SNARKs. One powerful method which can be used to construct fully homomorphic signatures is via CS-Proofs [Mic94] or, more generally, *succinct non-interactive arguments of knowledge for* **NP** (SNARKs) [BCCT12, BCCT13, BCI$^+$13, GGPR13, PHGR13, BSCG$^+$13]. This primitive allows us to non-interactively create a short "argument" π for any **NP** statement so that π

[1] The work of Catalano et al. [CFGN14] also proposes a framework of how to get fully homomorphic MACs with verification queries by relying on multilinear maps with very strict efficiency requirements. Unfortunately, we currently do not have any candidates for such maps.

proves “knowledge” of the corresponding witness. The length of π is bounded by some fixed polynomial in the security parameter and is independent of the statement/witness size. The complexity of verifying π only depends on the size of the statement (but not the witness). Using SNARKs, we can authenticate the output $y = g(x)$ of any computation g over any signed data x (under any standard signature scheme) by creating a short argument $\pi_{g,y}$ that proves the knowledge of “data x along with valid signature of x, such that $g(x) = y$”.

One advantage of the SNARK-based scheme is that a signature can be verified very efficiently, independently of the complexity of the computation g being verified (as long as g has a short Turing-Machine description). This is not the case for the other homomorphic MAC/Signature schemes from the literature or for the results of this work. Unfortunately, constructing SNARKs, even without a “knowledge” requirement, is known to require the use of *non-standard* assumptions [GW11]. The additional requirement of (non black-box) knowledge extraction makes SNARKs even more problematic and is unlikely to be possible in its full generality [BCPR14]. Known SNARK constructions are either based on the random-oracle heuristic *and* the use of heavy PCP machinery, or on various “knowledge of exponent” assumptions and light-weight PCP variants.

Other Related Work. Several other works consider the related problem of *delegating computation* to a remote server while maintaining the ability to efficiently verify the result [GKR08, GGP10, CKV10, AIK10, BGV11, CKLR11, PST13, KRR14]. In this scenario, the server needs to convince the user that $g(x) = y$, where the user knows the function g, the input x and the output y, but does not want to do the *work* of computing $g(x)$. In contrast, in our scenario the verifier only knows g, y (and a verification key) but does *not* know the previously authenticated data x, which may be huge. The latter scenario was considered by Chung et al. [CKLR11] in the context of *memory delegation*, where the client can also update the data on the server. However, all these solutions require some interaction between the client and server, and therefore do not yield homomorphic signatures.

1.2 Our Results

In this work, we construct the first *leveled fully homomorphic signature* schemes that can evaluate arbitrary circuits over signed data, where only the maximal depth d of the circuit needs to be fixed a priori. The size of the evaluated signature grows polynomially in d, but is otherwise independent of the circuit size. Our solutions are based on the hardness of the *small integer solution* (SIS) problem, which is in turn implied by the worst-case hardness of standard lattice problems [Ajt96].

We get a “bounded data” scheme in the standard model, where the total amount of data that can be signed needs to be bounded during setup and the size of the public parameters grows depending on this bound. In the random-oracle model, we get rid of this caveat and get a “multi data” scheme with short public parameters where the user can sign many different data-sets of arbitrary sizes. One can think of our results as extending those of Boneh and Freeman [BF11a] in two main dimensions:

- *Level of Homomorphism*: Most importantly, our scheme provides a significantly higher level of homomorphism. In the scheme of Boneh and Freeman, the signature size is (polynomially) related to the degree k of the polynomial being evaluated, whereas in our scheme it is (polynomially) related to the depth d of the evaluated circuit. In general, a circuit of depth d can translate into a polynomial of degree as high as $k = 2^d$, and therefore this is an exponential improvement.

- *Assumptions & Simplicity*: The scheme of Boneh and Freeman makes use of ideal lattices, whereas our scheme is based on standard lattice problems. Beyond improving on assumptions, avoiding ideal lattices also makes our scheme conceptually simpler.

Composition. Our schemes also allow composition of several different computations over signed data. One evaluator can compute some functions $y_1 = h_1(x), \ldots, y_\ell = h_\ell(x)$ over signed data x and publish the homomorphically computed signatures $\sigma_{h_1,y_1},\ldots,\sigma_{h_\ell,y_\ell}$. A second evaluator can then come and perform an additional computation $y^* = g(y_1,\ldots,y_\ell)$ on the output of the previous computation and combine the signatures $\sigma_{h_1,y_1},\ldots,\sigma_{h_\ell,y_\ell}$ into $\sigma_{g\circ\bar{h},y^*}$ which certifies y^* as the output of the composed computation $(g\circ\bar{h})(x) \stackrel{\text{def}}{=} g(h_1(x),\ldots,h_\ell(x))$. The second evaluator does not need to know the original data x or the original signatures. This can continue as long as the total computation is of bounded depth d.

Context Hiding. Our schemes can also be made *context hiding* so that a signature $\sigma_{g,y}$ does not reveal any additional information about the underlying data x beyond the output $y = g(x)$. We show how to achieve this statistically without relying on any additional assumptions.

1.3 Our Techniques

Our constructions are modular and, as a building block of potentially independent interest, we present a new primitive called a *homomorphic trapdoor function* (HTDF). We now give a high-level overview of our techniques. We start with the notion of HTDFs, then show how to construct homomorphic signatures from HTDFs, and finally show how to construct HTDFs from the SIS problem.

Homomorphic Trapdoor Functions (HTDF). An HTDF consists of a function $f_{pk,x}(\cdot)$ defined via a *public key pk* and an *index* $x\in\{0,1\}$. We can generate pk together with a trapdoor sk that allows us to invert the function $f_{pk,x}$ for any index x.[2] Given some values

$$u_1, x_1, v_1 = f_{pk,x_1}(u_1),\ldots,u_N, x_N, v_N = f_{pk,x_N}(u_N)$$

and a circuit $g\ :\ \{0,1\}^N \to \{0,1\}$, we can homomorphically (and deterministically) compute an input u^* (from x_i and u_i) and an output v^* (only from v_i) such that:

$$u^* := \mathsf{Eval}^{in}(g,(x_1,u_1)\ldots,(x_N,u_N))\ ,\ v^* := \mathsf{Eval}^{out}(g,v_1,\ldots,v_N) \quad\Rightarrow\quad f_{pk,g(x_1,\ldots,x_N)}(u^*) = v^*.$$

For security, we simply require that the HTDF is *claw-free*: given pk, it should be hard to come up with inputs u_0,u_1 such that $f_{pk,0}(u_0) = f_{pk,1}(u_1)$.[3,4]

Signatures from HTDFs in Standard Model. In the standard model, we construct a "bounded data" signature scheme where the setup procedure knows some bound N on the size of the data-set that will be signed and the size of the public parameters can depend on N. The public parameters $\mathsf{prms} = (v_1,\ldots,v_N)$ consist of N random outputs of the HTDF. Each user chooses a pubic/secret key pair (pk, sk) for an HTDF, which also serves as the key pair for the signature scheme. To sign some data $(x_1,\ldots,x_N)$ the user simply finds inputs u_i such that $f_{pk,x_i}(u_i) = v_i$ by using sk to invert v_i. We think of each u_i as a signature that ties x_i to its position i.

Given the values $x_1,\ldots,x_N$, the signatures $u_1,\ldots,u_N$, and a function $g\ :\ \{0,1\}^N \to \{0,1\}$, anybody can homomorphically compute a signature $u^*_{g,y} := \mathsf{Eval}^{in}(g,(x_1,u_1),\ldots,(x_N,u_N))$ which certifies $y = g(x_1,\ldots,x_N)$ as the output of the computation g. To verify the tuple $(g,y,u^*_{g,y})$, the verifier computes $v^* := \mathsf{Eval}^{out}(g,v_1,\ldots,v_N)$ and checks $f_{pk,y}(u^*_{g,y}) \stackrel{?}{=} v^*$. Notice that verification only depends on the public parameters but not on the data x.

[2] The function $f_{pk,x}$ may not be one-to-one, in which case we require that the inverting procedure comes up with a pre-image from the correct distribution.

[3] Our full definition/construction, allows x to come from a larger domain than just a single bit and considers arithmetic rather than just boolean circuits.

[4] One can also think of this primitive as a trapdoor homomorphic commitment. The values v_i are commitments to the bits x_i with opening u_i. One can homomorphically evaluate a function g over the commitments/openings to derive a commitment v^* to the bit $g(x_1,\ldots,x_N)$ with opening u^*. The trapdoor allows us to open any commitment v to any bit x.

The scheme satisfies selective security, where we assume that data-set $x_1, \dots, x_N$ is chosen by the attacker before seeing the public parameters. In the reduction, instead of choosing v_i randomly, we choose a random input u_i and set $v_i := f_{pk,x_i}(u_i)$. This makes it easy for the reduction to generate signatures u_i for the message bits x_i without knowing sk. Assume that the attacker produces a forgery (g, y', u') such that $y' \neq \{y := g(x_1, \dots, x_N)\}$ but $f_{pk,y'}(u') = \{v^* := \mathsf{Eval}^{out}(g, v_1, \dots, v_N)\}$. Then the reduction can compute $u = \mathsf{Eval}^{in}(g, (x_1, u_1), \dots, (x_N, u_N))$ so that $f_{pk,y}(u) = v^* = f_{pk,y'}(u')$. Therefore, the reduction produces values (u, u') which break the claw-free security of the HTDF.

Signatures from HTDFs in the RO Model. In the random-oracle model, we can get rid of the large public parameters and also achieve full rather than selective security. Moreover, following Boneh and Freeman [BF11a], we also allow the user to sign multiple different data-sets under different *labels* τ (e.g., τ can correspond to a "file name"), where verifier must simply know the label of the data-set on which the computation was supposed to be performed. Intuitively, instead of publishing the values v_i in the public parameters as in the previous scheme, we can derive them via hashing. Concretely, to sign some data-set of size N and with a label τ, we compute the values $\{\ v_i := H(\tau, r, i)\}_{i \in [N]}$, where r is some randomness generated by the signing procedure and included as part of the signature, and H is a random oracle. With this scheme, there is no a priori bound on the number of data-sets that can be signed or their sizes.

Constructing HTDFs. We construct HTDFs based on the SIS problem. The SIS problem states that, for a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ it should be hard to come up with a "short" non-zero vector $\mathbf{u} \in \mathbb{Z}_q^m$, such that $\mathbf{A} \cdot \mathbf{u} = \mathbf{0}$. However, there is a way to generate $\mathbf{A}$ along with a trapdoor td that makes this easy and, more generally, for any matrix $\mathbf{V} \in \mathbb{Z}_q^{n \times m}$, the trapdoor can be used to sample a "short" matrix $\mathbf{U} \in \mathbb{Z}_q^{m \times m}$ such that $\mathbf{AU} = \mathbf{V}$. There is also a public matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ with some special structure (not random) for which everyone has the above capability without needing a trapdoor.

Our HTDF consists of $pk = \mathbf{A}$ and $sk = \mathsf{td}$ as described above. We define $f_{pk,x}(\mathbf{U}) \stackrel{\text{def}}{=} \mathbf{AU} - x \cdot \mathbf{G}$, but restrict the domain to "short" matrices $\mathbf{U}$. Assume one can find a claw consisting of "short" matrices $\mathbf{U}_0, \mathbf{U}_1$ such that $f_{pk,0}(\mathbf{U}_0) = f_{pk,1}(\mathbf{U}_1)$. Then we can solve $\mathbf{A}(\mathbf{U}_1 - \mathbf{U}_0) = \mathbf{G}$. Since we can also sample a "short" $\mathbf{r}$ such that $\mathbf{G} \cdot \mathbf{r} = \mathbf{0}$ we get a "short" solution $\mathbf{u} := (\mathbf{U}_1 - \mathbf{U}_0) \cdot \mathbf{r}$ such that $\mathbf{A} \cdot \mathbf{u} = \mathbf{0}$, which breaks the SIS problem.[5] Next, we show how to perform homomorphic operations on this HTDF.

Homomorphic Operations. Let $\mathbf{U}_1, \mathbf{U}_2 \in \mathbb{Z}_q^{m \times m}$ be "short" matrices and

$$\mathbf{V}_1 = f_{pk,x_1}(\mathbf{U}_1) = \mathbf{AU}_1 - x_1 \cdot \mathbf{G} \ \ , \ \ \mathbf{V}_2 = f_{pk,x_2}(\mathbf{U}_2) = \mathbf{AU}_2 - x_2 \cdot \mathbf{G}$$

Addition. Firstly, it is very easy to perform homomorphic addition (over $\mathbb{Z}_q$). We can simply set: $\mathbf{V}^* = \mathbf{V}_1 + \mathbf{V}_2$ and $\mathbf{U}^* = \mathbf{U}_1 + \mathbf{U}_2$ and get:

$$f_{pk,x_1+x_2}(\mathbf{U}^*) = \mathbf{AU}^* - (x_1 + x_2)\mathbf{G} = (\mathbf{AU}_1 - x_1 \cdot \mathbf{G}) + (\mathbf{AU}_1 - x_1 \cdot \mathbf{G}) = \mathbf{V}^*$$

Multiplication. Homomorphic multiplication (over $\mathbb{Z}_q$) is slightly more complex and relies on a technique that was recently used by Boneh et al. [BGG+14] to get key-homomorphic encryption and attribute-based encryption (ABE). First we (deterministically) find a short matrix $\mathbf{R}$ such that $\mathbf{GR} = -\mathbf{V}_1$ using the special structure of $\mathbf{G}$. Then we set $\mathbf{V}^* := \mathbf{V}_2\mathbf{R}$ and $\mathbf{U}^* := x_2\mathbf{U}_1 + \mathbf{U}_2\mathbf{R}$. This gives:

$$\begin{aligned} f_{pk,x_1 \cdot x_2}(\mathbf{U}^*) &= \mathbf{AU}^* - (x_1 \cdot x_2)\mathbf{G} = \mathbf{A}(x_2\mathbf{U}_1 + \mathbf{U}_2\mathbf{R}) - (x_1 \cdot x_2)\mathbf{G} \\ &= x_2(\mathbf{AU}_1 - x_1\mathbf{G}) + \mathbf{AU}_2\mathbf{R} = x_2\mathbf{V}_1 + (\mathbf{V}_2 + x_2\mathbf{G})\mathbf{R} = x_2\mathbf{V}_1 + \mathbf{V}_2\mathbf{R} - x_2\mathbf{V}_1 \\ &= \mathbf{V}_2\mathbf{R} = \mathbf{V}^*. \end{aligned}$$

[5] This conveys the main intuition, but we must also argue that $\mathbf{u} \neq \mathbf{0}$. See the full proof for details.

We define the *noise level* of a matrix $\mathbf{U}$ to be the maximal size (in absolute value) of any entry in the matrix. The noise level grows as we perform homomorphic operations. Intuitively, if the inputs to the operation have noise-level β then homomorphic addition just doubles the noise level to 2β, while multiplication of "small" values $x_1, x_2 \in \{0,1\}$ multiplies the noise level of by some fixed multiplier γ to $\gamma\beta$. Therefore, when evaluating a boolean circuit of depth d, the noise level grows as γ^d. [6] Since the modulus must satisfy $q \gg \gamma^d$ for security, the level of homomorphism d is fixed ahead of time, during the setup of the scheme. The overall efficiency degrades *polynomially* with d.

2 Preliminaries

Basic Notation. For an integer N, we let $[N] \overset{\text{def}}{=} \{1, \ldots, N\}$. For a distribution X we use the notation $x \leftarrow X$ to denote the process of sampling a random value according to the distribution. For a set $\mathcal{X}$ we use the notation $x \overset{\$}{\leftarrow} \mathcal{X}$ to denote the process of choosing x uniformly at random from $\mathcal{X}$. For a distribution or a randomized algorithm X, we will say *"for any $x \in X$"* as shorthand to mean *"for any x in the support of X"*. Throughout, we let λ denote the security parameter. We say that a function $f(\lambda)$ is negligible, denoted $f(\lambda) = \mathsf{negl}(\lambda)$, if $f(\lambda) = O(\lambda^{-c})$ for ever constant $c > 0$. We say that a function $f(\lambda)$ is polynomial, denoted $f(\lambda) = \mathsf{poly}(\lambda)$ if $f(\lambda) = O(\lambda^c)$ for some constant $c > 0$.

Entropy and Statistical Distance. For random variables X, Y with support $\mathcal{X}, \mathcal{Y}$ respectively, we define the *statistical distance* $\mathbf{SD}(X,Y) \overset{\text{def}}{=} \frac{1}{2}\sum_{u \in \mathcal{X} \cup \mathcal{Y}} |\Pr[X = u] - \Pr[Y = u]|$. We say that two ensembles of random variables $X = \{X_\lambda\}, Y = \{Y_\lambda\}$ are *statistically close*, denoted by $X \overset{\text{stat}}{\approx} Y$, if $\mathbf{SD}(X_\lambda, Y_\lambda) = \mathsf{negl}(\lambda)$. The *min-entropy* of a random variable X, denoted as $\mathbf{H}_\infty(X)$, is defined as $\mathbf{H}_\infty(X) \overset{\text{def}}{=} -\log(\max_x \Pr[X = x])$. The *(average-)conditional min-entropy* of a random variable X conditioned on a correlated variable Y, denoted as $\mathbf{H}_\infty(X|Y)$, is defined as

$$\mathbf{H}_\infty(X|Y) \overset{\text{def}}{=} -\log\left(\underset{y \leftarrow Y}{\mathbf{E}}\left[\max_x \Pr[X = x | Y = y]\right]\right).$$

The optimal probability of an unbounded adversary guessing X given the correlated value Y is $2^{-\mathbf{H}_\infty(X|Y)}$.

Lemma 2.1 ([DORS08]). *Let X, Y be arbitrarily random variables where the support of Y lies in $\mathcal{Y}$. Then $\mathbf{H}_\infty(X|Y) \geq \mathbf{H}_\infty(X) - \log(|\mathcal{Y}|)$.*

2.1 Background on Lattices and the SIS Problem

We review some of the needed results and notation for the SIS problem and lattice-based cryptography. We abstract out many low-level details which are not absolutely crucial for this paper.

Notation. For any integer $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers modulo q. We represent elements of $\mathbb{Z}_q$ as integers in the range $(-q/2, q/2]$ and define the absolute value $|x|$ of $x \in \mathbb{Z}_q$ by taking its representative in this range. For a vector $\mathbf{u} \in \mathbb{Z}_q^n$ we write $||\mathbf{u}||_\infty \leq \beta$ if each entry u_i in $\mathbf{u}$ satisfies $|u_i| \leq \beta$. Similarly, for a matrix $\mathbf{U} \in \mathbb{Z}_q^{n \times m}$ we write $||\mathbf{U}||_\infty \leq \beta$ if each entry $u_{i,j}$ in $\mathbf{U}$ satisfies $|u_{i,j}| \leq \beta$.

The SIS Problem. Let n, m, q, β be integer parameters. In the $\mathsf{SIS}(n,m,q,\beta)$ problem, the attacker is given a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and her goal is to find a vector $\mathbf{u} \in \mathbb{Z}_q^m$ with $\mathbf{u} \neq \mathbf{0}$ and

[6] This is a crucial difference between our scheme and that of Boneh and Freeman [BF11a], where multiplication raises the noise level from β to β^2. In that scheme, a circuit of depth d could raise the noise level as high as β^{2^d}.

$||\mathbf{u}||_\infty \le \beta$ such that $\mathbf{A} \cdot \mathbf{u} = \mathbf{0}$.[7] For parameters $n = n(\lambda), m = m(\lambda), q = q(\lambda), \beta = \beta(\lambda)$ defined in terms of the security parameter λ, the $\mathsf{SIS}(n, m, q, \beta)$ hardness assumption states any PPT attacker $\mathcal{A}$ we have

$$\Pr\Big[\ \mathbf{A} \cdot \mathbf{u} = \mathbf{0}\ \wedge\ ||\mathbf{u}||_\infty \le \beta(\lambda)\ \wedge\ \mathbf{u} \ne \mathbf{0}\ \ :\ \ \mathbf{A} \overset{\$}{\leftarrow} \mathbb{Z}_{q(\lambda)}^{m(\lambda)\times n(\lambda)}, \mathbf{u} \leftarrow \mathcal{A}(1^\lambda, \mathbf{A})\ \Big] \le \mathsf{negl}(\lambda).$$

The SIS problem is known to be as hard as certain worst-case problems (e.g., SIVP) in standard lattices [Ajt96, Mic04, MR07, MP13]. It is is also implied by the hardness of *learning with errors* (LWE). See cited works for exact details of parameters. In this work, we will need to rely on the SIS assumption with super-polynomial β. In particular, we will assume that for any $\beta = 2^{\mathsf{poly}(\lambda)}$ there are some $n = \mathsf{poly}(\lambda)$, $q = 2^{\mathsf{poly}(\lambda)}$ (clearly, $q > \beta$) such that for all $m = \mathsf{poly}(\lambda)$ the $\mathsf{SIS}(n, m, q, \beta)$ hardness assumption holds. The above parameters translate to assuming hardness of worst-case lattice problems with sub-exponential approximation factors, which is widely believed to hold.

Lattice Trapdoors. Although solving the SIS problem for a random matrix $\mathbf{A}$ is believed to be hard, there is a way to sample a random matrix $\mathbf{A}$ with a trapdoor that makes this problem easy. Moreover, there are some fixed (non-random) matrices $\mathbf{G}$ for which SIS is easy to solve. We review the known results about such trapdoor in the following lemma (ignoring all details of implementation which aren't strictly necessary for us), following a similar presentation in [BGG+14].

Lemma 2.2 ([Ajt99, GPV08, AP09, MP12])**.** *There exist efficient algorithms* $\mathsf{TrapGen}$, SamPre, Sam *such that the following holds. Given integers* $n \ge 1, q \ge 2$ *there exists some* $m^* = m^*(n, q) = O(n \log q)$, $\beta_{sam} = \beta_{sam}(n, q) = O(n\sqrt{\log q})$ *such that for all* $m \ge m^*$ *and all* k *(polynomial in* n*) we have:*

1. $\mathbf{U} \leftarrow \mathsf{Sam}(1^m, 1^k, q)$ *samples a matrix* $\mathbf{U} \in \mathbb{Z}_q^{m\times k}$ *which satisfies* $||\mathbf{U}||_\infty \le \beta_{sam}$ *(with probability 1).*
2. *We have the statistical indistinguishability requirements:*

$$\mathbf{A} \overset{\text{stat}}{\approx} \mathbf{A}' \qquad \textit{and} \qquad (\mathbf{A}, \mathsf{td}, \mathbf{U}, \mathbf{V}) \overset{\text{stat}}{\approx} (\mathbf{A}, \mathsf{td}, \mathbf{U}', \mathbf{V}')$$

where $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, $\mathbf{A}' \overset{\$}{\leftarrow} \mathbb{Z}_q^{n\times m}$ *and* $\mathbf{U} \leftarrow \mathsf{Sam}(1^m, 1^k, q)$, $\mathbf{V} := \mathbf{A} \cdot \mathbf{U}$, $\mathbf{V}' \overset{\$}{\leftarrow} \mathbb{Z}_q^{n\times k}$, $\mathbf{U}' \leftarrow \mathsf{SamPre}(\mathbf{A}, \mathbf{V}', \mathsf{td})$. *The statistical distance is negligible in* n. *Moreover, we guarantee that any* $\mathbf{U}' \in \mathsf{SamPre}(\mathbf{A}, \mathbf{V}', \mathsf{td})$ *always satisfies* $\mathbf{A}\mathbf{U}' = \mathbf{V}'$ *and* $||\mathbf{U}'||_\infty \le \beta_{sam}$.

3. *Given* n, m, q *and* k *as above, there is an efficiently and deterministically computable matrix* $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ *and a deterministic algorithm* $\mathbf{R} = \mathsf{InvPub}(\mathbf{G}, \mathbf{V})$ *which takes the input* $\mathbf{V} \in \mathbb{Z}_q^{n\times k}$ *and outputs* $\mathbf{R} \in \{0, 1\}^{m\times k}$ *such that* $\mathbf{G} \cdot \mathbf{R} = \mathbf{V}$.

3 Homomorphic Trapdoor Functions

A homomorphic trapdoor function allows us to take values $\{u_i, x_i, v_i = f_{pk,x_i}(u_i)\}_{i\in[N]}$ and create an input u^* (depending on u_i, x_i) and an output v^* (depending only on v_i) such that $f_{pk,g(x_1,\ldots,x_N)}(u^*) = v^*$. We now give a formal definition.

3.1 Definition

A *homomorphic trapdoor function (HTDF)* consists of the following five polynomial-time algorithms $(\mathsf{KeyGen}, f, \mathsf{Inv}, \mathsf{Eval}^{in}, \mathsf{Eval}^{out})$ with syntax:

[7] Often, the SIS problem is stated with ℓ_2 norm rather than ℓ_∞ norm. It's clear that the two versions are equivalent up to some small losses of parameters. Therefore, we choose to rely on the ℓ_∞ norm for simplicity.

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$: a key generation procedure.
  The security parameter λ defines the *index space* $\mathcal{X}$, the *input space* $\mathcal{U}$, and the *output space* $\mathcal{V}$ and some efficiently samplable *input distribution* $D_\mathcal{U}$ over $\mathcal{U}$. We require that membership in the sets $\mathcal{U}, \mathcal{V}, \mathcal{X}$ can be efficiently tested and that one can efficiently sample uniformly at random from $\mathcal{V}$.
- $f_{pk,x} : \mathcal{U} \to \mathcal{V}$: a deterministic function indexed by $x \in \mathcal{X}$ and pk.
- $\mathsf{Inv}_{sk,x} : \mathcal{V} \to \mathcal{U}$: a probabilistic inverter indexed by $x \in \mathcal{X}$ and sk.
- $u^* = \mathsf{Eval}^{in}_{pk}(g, (x_1, u_1), \ldots, (x_\ell, u_\ell))$, $v^* = \mathsf{Eval}^{out}_{pk}(g, v_1, \ldots, v_\ell)$ are deterministic input/output homomorphic evaluation algorithms. The algorithms take as input some function $g : \mathcal{X}^\ell \to \mathcal{X}$ and values $x_i \in \mathcal{X}$, $u_i \in \mathcal{U}$, $v_i \in \mathcal{V}$. The outputs are $u^* \in \mathcal{U}$ and $v^* \in \mathcal{V}$.[8]

Note that we do not require $f_{pk,x}(\cdot)$ to be an injective function. Indeed, it will not be in our construction.

Correctness of Homomorphic Evaluation. Let $(pk, sk) \in \mathsf{KeyGen}(1^\lambda)$,[9] $x_1, \ldots, x_\ell \in \mathcal{X}$, $g : \mathcal{X}^\ell \to \mathcal{X}$ and $y := g(x_1, \ldots, x_\ell)$. Let $u_1, \ldots, u_\ell \in \mathcal{U}$ and set $v_i := f_{pk,x_i}(u_i)$ for $i \in [\ell]$. We require:

$$u^* := \mathsf{Eval}^{in}_{pk}(g, (x_1, u_1), \ldots, (x_\ell, u_\ell)) \;\; , \;\; v^* := \mathsf{Eval}^{out}_{pk}(g, v_1, \ldots, v_\ell) \quad \Rightarrow \quad u^* \in \mathcal{U} \; , \; f_{pk,y}(u^*) = v^*.$$

Relaxation: In a *leveled* fully homomorphic scheme, each input $u_i \in \mathcal{U}$ will have some associated "noise-level" $\beta_i \in \mathbb{R}$. The initial samples from the input-distribution $D_\mathcal{U}$ have some "small" noise-level β_{init}. The noise-level β^* of the homomorphically computed input u^* depends on the noise-levels β_i of the inputs u_i, the function g and the indices x_i. If the noise level β^* of u^* exceeds some threshold $\beta^* > \beta_{max}$, then the above correctness need not hold. This will limit the type of functions that can be evaluated. A function g is *admissible* on the values $x_1, \ldots, x_\ell$ if, whenever the inputs u_i have noise-levels $\beta_i \leq \beta_{init}$, then $u^* := \mathsf{Eval}^{in}_{pk}(g, (x_1, u_1), \ldots, (x_\ell, u_\ell))$ will have noise-level $\beta^* \leq \beta_{max}$.

Distributional Equivalence of Inversion. We require the following statistical indistinguishability:

$$(pk, sk, x, u, v) \overset{\text{stat}}{\approx} (pk, sk, x, u', v')$$

where $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $x \in \mathcal{X}$ can be an arbitrary random variable that depends on (pk, sk), $u \leftarrow D_\mathcal{U}$, $v := f_{pk,x}(u)$, $v' \xleftarrow{\$} \mathcal{V}$, $u' \leftarrow \mathsf{Inv}_{sk,x}(v')$.

HTDF Security. We now define the security of HTDFs. Perhaps the most natural security requirement would be *one-wayness*, meaning that for a random $v \leftarrow \mathcal{V}$ and any $x \in \mathcal{X}$ it should be hard to find a pre-image $u \in \mathcal{U}$ such that $f_{pk,x}(u) = v$. Instead, we will require a stronger property which is similar to *claw-freeness*. In particular, it should be difficult to find $u, u' \in \mathcal{U}$ and $x \neq x' \in \mathcal{X}$ such that $f_{pk,x}(u) = f_{pk,x'}(u')$. Formally, we require that for any PPT attacker $\mathcal{A}$ we have:

$$\Pr\left[\begin{array}{c} f_{pk,x}(u) = f_{pk,x'}(u') \\ u, u' \in \mathcal{U}, \quad x, x' \in \mathcal{X} \quad , x \neq x' \end{array} \;\middle|\; \begin{array}{c} (pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (u, u', x, x') \leftarrow \mathcal{A}(1^\lambda, pk) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

3.2 Construction: Basic Algorithms and Security

We begin by describing the basic HTDF algorithms for key-generation, computing the function $f_{pk,x}$, and inverting it using sk. We prove the security of the scheme. Then, in Section 3.3 we show how to perform homomorphic operations.

[8] More precisely, g is a *function description* in some specified format. In our case, this will always be either a boolean or an arithmetic circuit. For simplicity we often say "function g" but refer to a specific representation of the function.

[9] Recall, we use this as shorthand for "(pk, sk) in the support of $\mathsf{KeyGen}(1^\lambda)$".

Parameters. Our scheme will be defined by a flexible parameter $d = d(\lambda) = \mathsf{poly}(\lambda)$ which roughly determines the level of homomorphism. We choose parameters:

$$n \ , \ m \ , \ q \ , \ \beta_{SIS} \ , \ \beta_{max} \ , \ \beta_{init}$$

depending on λ and d. We do so by setting $\beta_{max} := 2^{\omega(\log \lambda)d}$, $\beta_{SIS} := 2^{\omega(\log \lambda)}\beta_{\max}$. Then choose an integer $n = \mathsf{poly}(\lambda)$ and a prime $q = 2^{\mathsf{poly}(\lambda)} > \beta_{SIS}$ as small as possible so that the $\mathsf{SIS}(n, m, q, \beta_{SIS})$ assumption holds for all $m = \mathsf{poly}(\lambda)$. Finally, let $m^* = m^*(n, q) := O(n \log q), \beta_{sam} := O(n\sqrt{\log q})$ be the parameters required by the trapdoor algorithms as in Lemma 2.2, and set $m = \max\{m^*, n \log q + \omega(\log \lambda)\} = \mathsf{poly}(\lambda)$ and $\beta_{init} := \beta_{sam} = \mathsf{poly}(\lambda)$. Note that $n, m, \log q$ all depend (polynomially) on λ, d.

Syntax. Let the algorithms $\mathsf{TrapGen}$, SamPre, Sam, and the matrix $\mathbf{G}$ be as defined in Lemma 2.2.

- Define $\mathcal{X} = \mathbb{Z}_q$ and $\mathcal{V} = \mathbb{Z}_q^{n\times m}$. Let $\mathcal{U} = \{\mathbf{U} \in \mathbb{Z}_q^{m\times m} \ : \ ||\mathbf{U}||_\infty \leq \beta_{max}\}$. We define the distribution $\mathbf{U} \leftarrow D_{\mathcal{U}}$ to sample $\mathbf{U} \leftarrow \mathsf{Sam}(1^m, 1^m, q)$ as in Lemma 2.2, so that $||\mathbf{U}||_\infty \leq \beta_{init}$.
- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$: Select $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$. Set $pk := \mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $sk = \mathsf{td}$.
- Define $f_{pk,x}(\mathbf{U}) \overset{\text{def}}{=} \mathbf{A} \cdot \mathbf{U} - x \cdot \mathbf{G}$. Note that, although the function f is well-defined on all of $\mathbb{Z}_q^{m\times m}$, we restrict the legal domain of f to the subset $\mathcal{U} \subseteq \mathbb{Z}_q^{m\times m}$.
- Define $\mathbf{U} \leftarrow \mathsf{Inv}_{sk,x}(\mathbf{V})$ to output $\mathbf{U} \leftarrow \mathsf{SamPre}(\mathbf{A}, \mathbf{V} + x \cdot \mathbf{G}, \mathsf{td})$.

We define the *noise-level* β of a value $\mathbf{U} \in \mathcal{U}$ as $\beta = ||\mathbf{U}||_\infty$. We note that all efficiency aspects of the scheme (run-time of procedures, sizes of keys/inputs/outputs, etc.) depend polynomially on λ and on the flexible parameter d.

Distributional Equivalence of Inversion. Let $(pk = \mathbf{A}, sk = \mathsf{td}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and let $x \in \mathcal{X}$ be an arbitrary random variable that depends on (pk, sk). Let $\mathbf{U} \leftarrow D_{\mathcal{U}}$, $\mathbf{V} = \mathbf{AU} - x \cdot \mathbf{G} = f_{pk,x}(\mathbf{U})$, $\mathbf{V}' \overset{\$}{\leftarrow} \mathcal{V}$, $\mathbf{U}' \leftarrow \{\mathsf{Inv}_{sk,x}(\mathbf{V}') = \mathsf{SamPre}(\mathbf{A}, \mathbf{V}' + x \cdot \mathbf{G}, \mathsf{td})\}$. Then we need to show:

$$(pk = \mathbf{A}, sk = \mathsf{td}, x, \mathbf{U}, \mathbf{V} = \mathbf{AU} - x\mathbf{G}) \overset{\text{stat}}{\approx} (pk = \mathbf{A}, sk = \mathsf{td}, x, \mathbf{U}', \mathbf{V}') \qquad (1)$$

Lemma 2.2, part (2) tells us that:

$$(\mathbf{A}, \mathsf{td}, \mathbf{U}, \mathbf{AU}) \overset{\text{stat}}{\approx} (\mathbf{A}, \mathsf{td}, \mathbf{U}', \mathbf{V}' + x \cdot \mathbf{G}) \qquad (2)$$

by noticing that $(\mathbf{V}' + x \cdot \mathbf{G})$ is just uniformly random. Equation (1) follows from (2) by applying the same function to both sides: append a sample x from the correct correlated distribution given $(\mathbf{A}, \mathsf{td})$ and subtract $x \cdot \mathbf{G}$ from the last component.

HTDF Security. We now prove the security of our HTDF construction under the SIS assumption.

Theorem 3.1. *Assuming the* $\mathsf{SIS}(n, m, q, \beta_{SIS})$*-assumption holds for the described parameter choices, the given scheme satisfies HTDF security.*

Proof. Assume that $\mathcal{A}$ is some PPT attacker that wins the HTDF security game for the above scheme with non-negligible probability. Let us modify the HTDF game so that, instead of choosing $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ and setting $pk := \mathbf{A}$ and $sk = \mathsf{td}$, we just choose $\mathbf{A} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n\times m}$ uniformly at random. Notice that $sk = \mathsf{td}$ is never used anywhere in the original HTDF game. Therefore, this modification is statistically indistinguishable by the security of $\mathsf{TrapGen}$ (see Lemma 2.2, part (2)). In particular, the probability of $\mathcal{A}$ winning the modified game remains non-negligible.

We now show that an attacker who wins the modified HTDF game can be used to solve the SIS problem. The reduction uses the challenge matrix $\mathbf{A}$ of the SIS problem as the public key $pk = \mathbf{A}$ and runs the attacker $\mathcal{A}$. Assume the attacker $\mathcal{A}$ wins the modified HTDF game with the values $\mathbf{U}, \mathbf{U}' \in \mathcal{U}$ and $x \neq x' \in \mathcal{X}$ such that $f_{pk,x}(\mathbf{U}) = f_{pk,x'}(\mathbf{U}')$. Let $\mathbf{U}^* := \mathbf{U}' - \mathbf{U}$ and $x^* = (x - x')$. Then:

$$f_{pk,x}(\mathbf{U}) = \mathbf{A}\mathbf{U} - x\mathbf{G} = \mathbf{A}\mathbf{U}' - x'\mathbf{G} = f_{pk,x'}(\mathbf{U}') \quad \Rightarrow \quad \mathbf{A}\mathbf{U}^* = x^*\mathbf{G} \tag{3}$$

Moreover, since $\mathbf{U}, \mathbf{U}' \in \mathcal{U}$, we have $||\mathbf{U}||_\infty, ||\mathbf{U}'||_\infty \leq \beta_{max}$ and therefore $||\mathbf{U}^*||_\infty \leq 2\beta_{max}$. Moreover, since $x \neq x'$, we have $x^* \neq 0$.

We now show that knowledge of a "small" $\mathbf{U}^*$ and some $x^* \neq 0$ satisfying the right hand side of equation (3) can be used to find a solution to the SIS problem. Sample $\mathbf{r} \overset{\$}{\leftarrow} \{0,1\}^m$, set $\mathbf{z} := \mathbf{A}\mathbf{r}$ and compute $\mathbf{r}' = \mathsf{InvPub}(\mathbf{G}, \mathbf{z}/x^*)$ so that $\mathbf{r}' \in \{0,1\}^m$ and $x^*\mathbf{G}\mathbf{r}' = \mathbf{z}$. Then

$$\mathbf{A}(\mathbf{U}^*\mathbf{r}' - \mathbf{r}) = (\mathbf{A}\mathbf{U}^*)\mathbf{r}' - \mathbf{A}\mathbf{r} = x^*\mathbf{G}\mathbf{r}' - \mathbf{A}\mathbf{r} = \mathbf{z} - \mathbf{z} = \mathbf{0}.$$

Therefore, letting $\mathbf{u} := \mathbf{U}^*\mathbf{r}' - \mathbf{r}$, we have $\mathbf{A}\mathbf{u} = \mathbf{0}$ and $||\mathbf{u}||_\infty \leq (2m+1)\beta_{max} \leq \beta_{SIS}$. It remains to show that $\mathbf{u} \neq \mathbf{0}$, or equivalently, that $\mathbf{r} \neq \mathbf{U}^*\mathbf{r}'$. We use an entropy argument to show that this holds with overwhelming probability over the random choice of $\mathbf{r}$, even if we fix some worst-case choice of $\mathbf{A}, \mathbf{U}^*, x^*$. Notice that $\mathbf{r} \overset{\$}{\leftarrow} \{0,1\}^m$ is chosen uniformly at random, but $\mathbf{r}'$ depends on $\mathbf{z} = \mathbf{A}\mathbf{r}$. Nevertheless $\mathbf{z}$ is too small to reveal much information about $\mathbf{r}$ and therefore cannot be used to predict $\mathbf{r}$. In particular

$$\mathbf{H}_\infty(\mathbf{r} \mid \mathbf{r}') \geq \mathbf{H}_\infty(\mathbf{r} \mid \mathbf{A}\mathbf{r}) \geq m - n\log q = \omega(\log\lambda)$$

where the first inequality follows since $\mathbf{r}'$ is chosen deterministically based on $\mathbf{z} = \mathbf{A}\mathbf{r}$, and the second inequality follows from Lemma 2.1. Therefore, $\Pr[\mathbf{r} = \mathbf{U}^* \cdot \mathbf{r}'] \leq 2^{m - n\log q} \leq \mathsf{negl}(\lambda)$. So, with overwhelming probability, whenever $\mathcal{A}$ wins the modified HTDF game, the reduction finds a valid solution to the $\mathsf{SIS}(n, m, q, \beta_{SIS})$-problem. This concludes the proof. □

3.3 Construction: Homomorphic Evaluation and Noise Growth

We now define the algorithms Eval^{in}, Eval^{out} with the syntax

$$\mathbf{U}^* := \mathsf{Eval}^{in}_{pk}(g, (x_1, \mathbf{U}_1), \ldots, (x_\ell, \mathbf{U}_\ell)) \quad , \quad \mathbf{V}^* := \mathsf{Eval}^{out}_{pk}(g, \mathbf{V}_1, \ldots, \mathbf{V}_\ell).$$

Our approach closely follows the techniques introduced by Boneh et al. [BGG+14]. As a basic building block, we consider homomorphic evaluation for certain base functions g which we think of as basic gates in an arithmetic circuit: *addition*, *multiplication*, *addition-with-constant* and *multiplication-by-constant*. These functions are complete and can be composed to evaluate an arbitrary aithmetic circuit. Let the matrices $\mathbf{U}_i$ have noise-levels bounded by β_i.

- Let $g(x_1, x_2) = x_1 + x_2$ be an *addition* gate. The algorithms $\mathsf{Eval}^{in}, \mathsf{Eval}^{out}$ respectively compute:

  $$\mathbf{U}^* := \mathbf{U}_1 + \mathbf{U}_2 \quad , \quad \mathbf{V}^* := \mathbf{V}_1 + \mathbf{V}_2.$$

  The matrix $\mathbf{U}^*$ has noise level $\beta^* \leq \beta_1 + \beta_2$. We remark that, in this case, the algorithm Eval^{in} ignores the values x_1, x_2.

- Let $g(x_1, x_2) = x_1 \cdot x_2$ be a *multiplication* gate. Let $\mathbf{R} = \mathsf{InvPub}(\mathbf{G}, -\mathbf{V}_1)$ so that $\mathbf{R} \in \{0,1\}^{m \times m}$ and $\mathbf{G}\mathbf{R} = -\mathbf{V}_1$. The algorithms $\mathsf{Eval}^{in}, \mathsf{Eval}^{out}$ respectively compute:

  $$\mathbf{U}^* := x_2 \cdot \mathbf{U}_1 + \mathbf{U}_2\mathbf{R} \quad , \quad \mathbf{V}^* := \mathbf{V}_2 \cdot \mathbf{R}.$$

  The matrix $\mathbf{U}^*$ has noise level $\beta^* \leq |x_2|\beta_1 + m\beta_2$. Note that the noise growth is asymmetric and the order of x_1, x_2 matters. To keep the noise level low, we require that $|x_2|$ is small.

- Let $g(x) = x + a$ be *addition-with-constant* gate, for the constant $a \in \mathbb{Z}_q$. The algorithms $\mathsf{Eval}^{in}, \mathsf{Eval}^{out}$ respectively compute:

  $$\mathbf{U}^* := \mathbf{U}_1 \quad , \quad \mathbf{V}^* := \mathbf{V}_1 - a \cdot \mathbf{G}.$$

  It's easy to see that the noise-level $\beta^* = \beta_1$ stays the same.

- Let $g(x) = a \cdot x$ be a *multiplication-by-constant* gate for the constant $a \in \mathbb{Z}_q$. We give two alternative methods that homomorphically compute g with different noise growth. In the first method, the algorithms $\mathsf{Eval}^{in}, \mathsf{Eval}^{out}$ respectively compute:

  $$\mathbf{U}^* := a \cdot \mathbf{U}_1 \quad , \quad \mathbf{V}^* := a \cdot \mathbf{V}_1.$$

  The noise level is $\beta^* = |a|\beta_1$, and therefore this method requires that a is small. In the second method, let $\mathbf{R} = \mathsf{InvPub}(\mathbf{G}, a \cdot \mathbf{G})$ so that $\mathbf{R} \in \{0,1\}^{m \times m}$ and $\mathbf{GR} = a \cdot \mathbf{G}$. The algorithms $\mathsf{Eval}^{in}, \mathsf{Eval}^{out}$ respectively compute:

  $$\mathbf{U}^* := \mathbf{U} \cdot \mathbf{R} \quad , \quad \mathbf{V}^* := \mathbf{V} \cdot \mathbf{R}.$$

  The noise level is $\beta^* \leq m \cdot \beta_1$, and is therefore independent of the size of a.

It is a simple exercise to check that, whenever the inputs $\mathbf{U}_i, \mathbf{V}_i$ satisfy $\mathbf{V}_i = f_{pk,x_i}(\mathbf{U}_i)$ then the above homomorphic evaluation procedures ensure that $f_{pk,g(x_1,\ldots,x_\ell)}(\mathbf{U}^*) = \mathbf{V}^*$. The above gate operations can be composed to compute any function g expressed as an arithmetic circuit. Therefore, the only limitation is the growth of the noise-level. In particular, if the noise-level of $\mathbf{U}^*$ is $\beta^* \geq \beta_{max}$ then $\mathbf{U}^* \notin \mathcal{U}$ is not a valid input.

Noise Growth and Bounded-Depth Circuits. The noise growth of the above homomorphic operations is fairly complex to describe in its full generality since it depends on the (size of) the inputs x_i, the order in which operations are performed etc. However, we can give bounds on the noise growth for the case of boolean circuits composed of NAND gates, and certain restricted arithmetic circuits.

Let g be a *boolean circuit of depth d composed of NAND gates* over inputs $x_i \in \{0,1\}$. For $x_1, x_2 \in \{0,1\}$ we can define an arithmetic-gate $\mathsf{NAND}(x_1, x_2) \overset{\text{def}}{=} 1 - x_1 \cdot x_2$. If U_1, U_2 have noise-levels $\leq \beta$, then $\mathbf{U}^* := \mathsf{Eval}^{in}_{pk}(\mathsf{NAND}, (x_1, \mathbf{U}_1), (x_2, \mathbf{U}_2))$ will have a noise-level $\beta^* \leq (m+1)\beta$. Therefore if we compute $\mathbf{U}^* := \mathsf{Eval}^{in}_{pk}(g, (x_1, \mathbf{U}_1), \ldots, (x_\ell, \mathbf{U}_\ell))$ and the inputs $\mathbf{U}_i$ have noise-levels β_{init}, then the noise-level of $\mathbf{U}^*$ will be $\beta^* \leq \beta_{init} \cdot (m+1)^d \leq 2^{O(\log \lambda) \cdot d} \leq \beta_{max}$. This show that, with the parameters we chose, any depth-d boolean circuit g is admissible over any choice of boolean indices $x_i \in \{0,1\}$.

More generally, let g be an *arithmetic circuit of depth d* consisting of fan-in-t addition gates, fan-in-2 multiplication gates, addition-with-constant, and multiplication-by-constant gates. Moreover, assume that for each fan-in-2 multiplication gate we are guaranteed that at least one input x_b is of size $|x_b| \leq p$, where $p = \mathsf{poly}(\lambda), t = \mathsf{poly}(\lambda)$ are some fixed polynomials in the security parameter. Evaluating each such gate increases the noise level by a multiplicative factor of at most $\max\{t, (p+m)\} = \mathsf{poly}(\lambda)$. Therefore, if inputs $\mathbf{U}_i$ to g have noise-levels β_{init}, then the noise-level of $\mathbf{U}^* := \mathsf{Eval}^{in}_{pk}(g, (x_1, \mathbf{U}_1), \ldots, (x_\ell, \mathbf{U}_\ell))$ is bounded by $\beta_{init} \cdot \max\{t, (p+m)\}^d \leq 2^{O(\log \lambda) \cdot d} \leq \beta_{max}$. This shows that any such computation is admissible.

We mention that both of the above analyses are overly restrictive/pessimistic and we may be able to compute some function with lower noise growth than suggested above.

4 Bounded-Data Homomorphic Signatures (Standard Model)

We now define and construct *bounded-data* homomorphic signatures. In such a scheme, the maximal amount of data that any user can sign is fixed ahead of time and the size of the public key (in our construction, it will only be the public parameters) can depend on this bound.

4.1 Definition

A *bounded-data homomorphic signature* scheme consists of poly-time algorithms ($\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Eval}$) with the following syntax.

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^N)$: Gets the security parameter λ and a data-size bound N. Generates a public/secret key pk, sk. The security parameter also defines the message space $\mathcal{X}$.
- $(\sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk}(x_1, \ldots, x_N)$: Signs some data $(x_1, \ldots, x_N) \in \mathcal{X}^N$.
- $\sigma^* = \mathsf{Eval}_{pk}(g, ((x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell)))$: Homomorphically computes a signature σ^*.
- $\mathsf{Verify}_{pk}(g, y, \sigma)$: Verifies that y is indeed the output of g by checking the signature σ.

Signing Correctness. Let $\mathsf{id}_i : \mathcal{X}^N \to \mathcal{X}$ be a canonical description of the function $\mathsf{id}_i(x_1, \ldots, x_N) \overset{\text{def}}{=} x_i$ (i.e., a circuit consisting of a single wire taking the i'th input to the output.) We require that any $(pk, sk) \in \mathsf{KeyGen}(1^\lambda, 1^N)$, any $(x_1, \ldots, x_N) \in \mathcal{X}^N$ and any $(\sigma_1, \ldots, \sigma_N) \in \mathsf{Sign}_{sk}(x_1, \ldots, x_N)$ must satisfy $\mathsf{Verify}_{pk}(\mathsf{id}_i, x_i, \sigma_i) = \mathtt{accept}$. In other words, σ_i certifies x_i as the i'th data item.

Evaluation Correctness. We require that for any $(pk, sk) \in \mathsf{KeyGen}(1^\lambda, 1^N)$, any $(x_1, \ldots, x_N) \in \mathcal{X}^N$ and any $(\sigma_1, \ldots, \sigma_N) \in \mathsf{Sign}_{sk}(x_1, \ldots, x_N)$ and any $g : \mathcal{X}^N \to \mathcal{X}$, we have:

$$\sigma^* := \mathsf{Eval}_{pk}(g, ((x_1, \sigma_1), \ldots, (x_N, \sigma_N)) \quad \Rightarrow \quad \mathsf{Verify}_{pk}(g, g(x_1, \ldots, x_N), \sigma^*) = \mathtt{accept}. \tag{4}$$

Moreover, we can compose the evaluation of several different functions. For any $h_1, \ldots, h_\ell$ with $h_i : \mathcal{X}^N \to \mathcal{X}$ and any $g : \mathcal{X}^\ell \to \mathcal{X}$ define the composition $(g \circ \bar{h}) : \mathcal{X}^N \to \mathcal{X}$ by $(g \circ \bar{h})(\bar{x}) = g(h_1(\bar{x}), \ldots, h_\ell(\bar{x}))$. We require that for any $(x_1, \ldots, x_\ell) \in \mathcal{X}^\ell$ and any $(\sigma_1, \ldots, \sigma_\ell)$:

$$\begin{array}{l} \{ \ \mathsf{Verify}_{pk}(h_i, x_i, \sigma_i) = \mathtt{accept} \ \}_{i \in [\ell]} \\ \sigma^* := \mathsf{Eval}_{pk}(g, (x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell)) \end{array} \quad \Rightarrow \quad \mathsf{Verify}_{pk}((g \circ \bar{h}), g(x_1, \ldots, x_\ell), \sigma^*) = \mathtt{accept}. \tag{5}$$

In other words, if the signatures σ_i certify x_i as the output of h_i, then σ^* certifies $g(x_1, \ldots, x_\ell)$ as the output of $g \circ \bar{h}$. Notice that (4) follows from (5) and the correctness of signing by setting $h_i \overset{\text{def}}{=} \mathsf{id}_i$.

Relaxing Correctness for Leveled Schemes. In a *leveled* fully homomorphic scheme, each signature σ_i will have some associated "noise-level" β_i. The initial signatures produced by $(\sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk}(x_1, \ldots, x_N)$ will have a "small" noise-level β_{init}. The noise-level β^* of the homomorphically computed signature $\sigma^* := \mathsf{Eval}_{pk}(g, ((x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell)))$ depends on the noise-levels β_i of the signatures σ_i, the function g and the messages x_i. If the noise level β^* of σ^* exceeds some threshold $\beta^* > \beta_{max}$, then the above correctness requirements need not hold. This will limit the type of functions that can be evaluated. A function g is *admissible* on the values $x_1, \ldots, x_\ell$ if, whenever the signatures σ_i have noise-levels $\beta_i \le \beta_{init}$, then σ^* will have noise-level $\beta^* \le \beta_{max}$.

Bounded-Data Security. By default, we will consider *selective security* for bounded-data homomorphic signatures, where the attacker chooses the data $x_1, \ldots, x_N$ to be signed before seeing pk. This is a natural security notion for the typical use-case where the user selects pk, sk and signs the data in one step and therefore the data will not depend on pk. We define the security of homomorphic signatures via the following game between an attacker $\mathcal{A}$ and a challenger:

- The attacker $\mathcal{A}(1^\lambda)$ chooses data $(x_1, \ldots, x_N) \in \mathcal{X}^*$ and sends it to the challenger.
- The challenger chooses $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^N)$, $(\sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk}(x_1, \ldots, x_N)$ and gives $pk, (\sigma_1, \ldots, \sigma_N)$ to $\mathcal{A}$.

- The attacker $\mathcal{A}$ chooses a function $g : \mathcal{X}^N \to \mathcal{X}$ and values y', σ'. Let $y := g(x_1, \ldots, x_N)$. The attacker wins if all of the following hold: (1) g is admissible on the values $x_1, \ldots, x_N$, (2) $y' \neq y$, and (3) $\mathsf{Verify}_{pk}(g, y', \sigma') = \mathtt{accept}$.

We require that for all PPT $\mathcal{A}$, we have $\Pr[\mathcal{A} \text{ wins }] \leq \mathsf{negl}(\lambda)$ in the above game.

Remark: Verification and Admissible Functions. We note that, under the above definition, the security of $\mathsf{Verify}_{pk}(g, y, \sigma)$ only holds when verifying a function g which is admissible on the signed values $x_1, \ldots, x_N$, but the verifier does not know these values. Therefore, we require some convention on the types of values x_i that the signer will sign and the type of functions g that the verifier is willing to verify to ensure that the function is admissible on the signed values. For example, our eventual construction ensures that if g is a boolean circuit of depth $\leq d$ then it is admissible on all boolean inputs with $x_i \in \{0,1\} \subseteq \mathcal{X}$. Therefore, by convention, we can restrict the signer to only sign values $x_i \in \{0,1\}$ and the verifier to only verify functions g that are boolean circuits of depth $\leq d$. Other combinations (e.g., $x_i \in \mathbb{Z}_q$ and g is an affine function) are also possible and therefore we leave this decision to the users of the scheme rather than its specification.

4.2 Construction

Let $\mathcal{F} = (\mathsf{KeyGen}^{HTDF}, f, \mathsf{Inv}, \mathsf{Eval}^{in}, \mathsf{Eval}^{out})$ be an HTDF with index-space $\mathcal{X}$, input space $\mathcal{U}$, output space $\mathcal{V}$ and an input distribution $D_{\mathcal{U}}$. We construct a signature scheme $\mathcal{S} = (\mathsf{KeyGen}^{sig}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Eval}^{sig})$ with message space $\mathcal{X}$ as follows.

- $(pk, sk) \leftarrow \mathsf{KeyGen}^{sig}(1^\lambda, 1^N)$: Choose $v_1, \ldots, v_N$ by sampling $v_i \overset{\$}{\leftarrow} \mathcal{V}$. Set $\mathsf{prms} = (v_1, \ldots, v_N)$. Choose $(pk', sk') \leftarrow \mathsf{KeyGen}^{HTDF}(1^\lambda)$ and set $pk = (\mathsf{prms}, pk')$, $sk = (\mathsf{prms}, sk')$.
- $(\sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk}(x_1, \ldots, x_N)$: Sample $u_i \leftarrow \mathsf{Inv}_{sk', x_i}(v_i)$ and set $\sigma_i := u_i$ for $i \in [N]$.
- $\sigma^* = \mathsf{Eval}^{sig}_{pk}(g, (x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell))$: This is the same as the $\mathsf{Eval}^{in}_{pk'}$ procedure of the HTDF.
- $\mathsf{Verify}_{pk}(g, y, \sigma)$: Compute $v^* := \mathsf{Eval}^{out}_{pk'}(g, v_1, \ldots, v_N)$. If $f_{pk', y}(\sigma) = v^*$ accept, else reject.

Remarks. (I) We can think of $\mathsf{prms} = (v_1, \ldots, v_N)$ as public parameters that can be fixed for all users of the scheme. Each user's individual public/secret key then only consists of the small values pk', sk'. (II) Although we describe the signing procedure as signing the values $x_1, \ldots, x_N$ in one shot, it's easy to see that we can also sign the values x_i completely independently (e.g., at different times) without needing to keep any state beyond knowing the index i by setting $\sigma_i \leftarrow \mathsf{Inv}_{sk', x_i}(v_i)$. (III) We note that if the function g only "touches" a small subset of the inputs $i \in [N]$ then the verification procedure only needs to read the corresponding values v_i from the public parameters. The run-time of the verification procedure can be sub-linear in N and only depends on the size of the circuit g (ignoring unused input wires).

Correctness and Security. It's easy to see that correctness of signing and correctness of (leveled) homomorphic evaluation for the signature scheme $\mathcal{S}$ follow from the correctness properties of the underlying (leveled) HTDF $\mathcal{F}$. In a leveled scheme, the noise-level of signatures $\sigma_i = u_i$ is just defined as its noise-level of the HTDF input u_i. The initial noise-level β_{init}, the maximal noise level β_{max}, and the set of admissible functions is the same in $\mathcal{S}$ and in $\mathcal{F}$. We are left to prove security.

Theorem 4.1. *Assuming $\mathcal{F}$ satisfies HTDF security, the signature scheme $\mathcal{S}$ satisfies bounded-data security of homomorphic signatures.*

Proof. Assume that an adversary $\mathcal{A}$ has a non-negligible advantage in the bounded-data security game with the scheme $\mathcal{S}$. In the game, the attacker $\mathcal{A}$ selects some data $x_1, \ldots, x_N \in \mathcal{X}$ and gets back $\mathsf{prms} = (v_1, \ldots, v_N)$, pk' and $\sigma_1, \ldots, \sigma_N$, where $(pk', sk') \leftarrow \mathsf{KeyGen}^{HTDF}(1^\lambda)$, $v_i \leftarrow \mathcal{V}$ and $\sigma_i = u_i \leftarrow \mathsf{Inv}_{sk',x_i}(v_i)$. Let us modify the game by choosing $u_i \leftarrow D_{\mathcal{U}}$ and setting $v_i := f_{pk',x_i}(u_i)$. This change is statistically indistinguishable by the "Distributional Equivalence of Inversion" property of the HTDF.[10] Therefore $\mathcal{A}$ wins the modified games with non-negligible probability.

We now give a polynomial-time reduction that takes any attacker $\mathcal{A}$ having a non-negligible advantage in the above modified game, and use it to break HTDF security of $\mathcal{F}$ with the same advantage. The reduction gets a challenge public key pk' and chooses the values u_i, v_i as in the modified game (without knowing sk') and gives these values to $\mathcal{A}$. Assume the attacker $\mathcal{A}$ wins the modified game by choosing some admissible function $g : \mathcal{X}^N \to \mathcal{X}$ on $x_1, \ldots, x_N$ and some values $y', \sigma' = u'$. Let $y := g(x_1, \ldots, x_N)$, $v^* := \mathsf{Eval}^{out}_{pk'}(g, v_1, \ldots, v_N)$, $u := \mathsf{Eval}^{in}_{pk'}(g, (x_1, \sigma_1), \ldots, (x_N, \sigma_N))$. Then, since the signature σ' verifies, we have $f_{pk',y'}(u') = v^*$. On the other hand, since g is an admissible function, the correctness of homomorphic evaluation ensures that $f_{pk',y}(u) = v^*$. Therefore, the values $u, u' \in \mathcal{U}$ and $y \neq y' \in \mathcal{X}$ satisfy $f_{pk',y}(u) = f_{pk',y'}(u')$, allowing the reduction to break HTDF security whenever $\mathcal{A}$ wins the modified game. □

5 Multi-Data Homomorphic Signatures (Random Oracle Model)

We now define and construct *multi-data* homomorphic signatures. In such a scheme, the signer can sign many different data-sets of arbitrary size. Each data-set is tied to some labels τ_i (e.g., the name of the data-set) and the verifier is assumed to know the label of the data-set over which he wishes to verify computation.

5.1 Definition

A *multi-data homomorphic signature* consists of the algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Eval})$ with the following syntax.

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$: produces a public verification key pk and a secret signing key sk. The security parameter λ defines some message alphabet $\mathcal{X}$.
- $(\sigma_\tau, \sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk}((x_1, \ldots, x_N), \tau)$: Signs some data $\bar{x} \in \mathcal{X}^*$ under a label $\tau \in \{0,1\}^*$.
- $\sigma^* = \mathsf{Eval}_{vk}(g, (x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell))$: Homomorphically computes the signature σ^*.
- $\mathsf{Verify}_{pk}(g, y, \tau, (\sigma_\tau, \sigma^*))$: Verifies that $y \in \mathcal{X}$ is indeed the output of the function g over the data signed with label τ.

Correctness. The correctness requirements are analogous to those of the bounded-data definition.

Correctness of Signing. We require that any $(pk, sk) \in \mathsf{KeyGen}(1^\lambda)$, any $(x_1, \ldots, x_N) \in \mathcal{X}^N$, any $\tau \in \{0,1\}^*$ and any $(\sigma_\tau, \sigma_1, \ldots, \sigma_N) \in \mathsf{Sign}_{sk}(x_1, \ldots, x_N, \tau)$ must satisfy $\mathsf{Verify}_{pk}(\mathsf{id}_i, x_i, \tau, (\sigma_\tau, \sigma_i)) = \mathtt{accept}$. In other words, (σ_τ, σ_i) certifies x_i as the i'th data item of the data with label τ.

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 $\tau \in \{0,1\}^*$ and any $\sigma_\tau, (\sigma_1, \ldots, \sigma_\ell)$:

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

In other words, if the signatures (σ_τ, σ_i) certify x_i as the outputs of functions h_i over the data labeled with τ, then (σ_τ, σ^*) certifies $g(x_1, \ldots, x_\ell)$ as the output of $g \circ \bar{h}$ over the data labeled with τ.

[10] Technically, this requires N hybrid arguments where we switch how each u_i, v_i is sampled one-by-one. In each hybrid, we rely on the fact that indistinguishability holds even given sk' to sample the rest of the values u_j, v_j.

Multi-Data Security. In the case of multi-data signatures, we will consider full adaptive security. We define the security via the following game between an attacker $\mathcal{A}$ and a challenger:

- The challenger chooses $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and gives pk to the attacker $\mathcal{A}$
- Signing Queries: The attacker $\mathcal{A}$ can ask an arbitrary number of signing queries. In each query j, the attacker chooses a fresh tag $\tau_j \in \{0,1\}^*$ which was never queried previously and a message $(x_{j,1}, \ldots, x_{j,N_j}) \in \mathcal{X}^*$. The challenger responds with
$$(\sigma_{\tau_j}, \sigma_{j,1}, \ldots, \sigma_{j,N_j}) \leftarrow \mathsf{Sign}_{sk}((x_{j,1}, \ldots, x_{j,N_j}), \tau_j).$$
- The attacker $\mathcal{A}$ chooses a circuit $g : \mathcal{X}^{N'} \to \mathcal{X}$ values $\tau, y', (\sigma'_\tau, \sigma')$. The attacker wins if $\mathsf{Verify}_{pk}(g, \tau, y', (\sigma'_\tau, \sigma')) = \mathtt{accept}$ and either:
  - *Type I forgery:* $\tau \neq \tau_j$ for any j, or $\tau = \tau_j$ for some j but $N' \neq N_j$.
  (i.e., No signing query with label τ was ever made or there is a mismatch between the size of the data signed under label τ and the arity of the function g.)
  - *Type II forgery:* $\tau = \tau_j$ for some j with corresponding message $x_{j,1}, \ldots, x_{j,N'}$ such that (a) g is admissible on $x_{j,1}, \ldots, x_{j,N'}$, and (b) $y' \neq g(x_{j,1}, \ldots, x_{j,N'})$.

We require that for all PPT $\mathcal{A}$, we have $\Pr[\mathcal{A} \text{ wins }] \leq \mathsf{negl}(\lambda)$ in the above game.

5.2 Construction

Let $\mathcal{F} = (\mathsf{KeyGen}^{HTDF}, f, \mathsf{Inv}, \mathsf{Eval}^{in}, \mathsf{Eval}^{out})$ be an HTDF with index-space $\mathcal{X}$, input space $\mathcal{U}$, output space $\mathcal{V}$ and an input distribution $D_{\mathcal{U}}$. Let $\mathcal{S}^{nh} = (\mathsf{KeyGen}^{nh}, \mathsf{Sign}^{nh}, \mathsf{Verify}^{nh})$ be any standard (not homomorphic) signature scheme. Let $H : \{0,1\}^* \to \mathcal{V}$ be a hash function modeled as a random oracle. We construct a multi-data homomorphic signature scheme $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Eval})$ with message space $\mathcal{X}$ as follows.

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$: Choose $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}^{HTDF}(1^\lambda)$ and $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}^{nh}(1^\lambda)$ set $pk = (pk_1, pk_2)$, $sk = (sk_1, sk_2)$.
- $(\sigma_\tau, \sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk}((x_1, \ldots, x_N), \tau)$: Sample $r \overset{\$}{\leftarrow} \{0,1\}^\lambda$ and $\rho \leftarrow \mathsf{Sign}^{nh}_{sk_2}((\tau, r, N))$. Set $\sigma_\tau = (r, N, \rho)$. For each $i \in [N]$, let $v_i := H(\tau, r, i)$ and set $\sigma_i \leftarrow \mathsf{Inv}_{sk_1, x_i}(v_i)$.
- $\sigma^* = \mathsf{Eval}^{sig}_{pk}(g, (x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell))$: This is the same as the $\mathsf{Eval}^{in}_{pk_1}$ procedure of the HTDF.
- $\mathsf{Verify}_{pk}(g, y, \tau, (\sigma_\tau, \sigma^*))$: Parse $\sigma_\tau = (r, N, \rho)$ and check $\mathsf{Verify}^{nh}_{pk_2}((\tau, r, N), \rho) = \mathtt{accept}$ (reject otherwise). For $i \in [N]$ compute $v_i := H(\tau, r, i)$ and $v^* := \mathsf{Eval}^{out}_{pk_1}(g, v_1, \ldots, v_N)$. If $f_{pk_2, y}(\sigma) = v^*$ accept, else reject.

Remarks. Note that, just like in the bounded-data construction, the run-time of the verification procedure can be sub-linear in N and only depends on the size of the circuit g (ignoring unused input wires). The above scheme can also easily allow homomorphic evaluation of functions that compute over multiple different data-sets with several different labels τ_i (each input-wire of the function g is associated with some label τ_i and some index $j \in [N_i]$). The verification procedure simply needs to know σ_{τ_i} for each of the the data-sets τ_i involved in the computation. For simplicity, we will not analyze this use-case further and stick with the definition which only allows computations over a single data-set.

Correctness and Security. It's easy to see that correctness of signing and correctness of (leveled) homomorphic evaluation for the signature scheme $\mathcal{S}$ follow from the correctness properties of the underlying (leveled) HTDF $\mathcal{F}$ and the non-homomorphic signature scheme $\mathcal{S}^{nh}$. In a leveled scheme, the noise-level of signatures $\sigma_i = u_i$ is just defined as its noise-level of the HTDF input u_i. The initial noise-level β_{init}, the maximal noise level β_{max}, and the set of admissible functions is the same in $\mathcal{S}$ and in $\mathcal{F}$. We are left to prove security.

Theorem 5.1. *Assuming $\mathcal{F}$ satisfies HTDF security, $\mathcal{S}^{nh}$ satisfies standard signature security, and H is modeled as a random oracle, the above scheme $\mathcal{S}$ satisfies multi-data security of homomorphic signatures.*

Proof. We prove the theorem via a sequence of games argument. Let $\mathcal{A}$ be a PPT attacker and, in each game i, let E_i be the event that the attacker wins the corresponding game.

- Let **Game** 0 be the multi-data security game. We wish to show that $\Pr[E_0] \le \mathsf{negl}(\lambda)$.

- Let **Game** 1 be a modified version of **Game** 0 where, in each signing query, we check that the value $r \leftarrow \{0,1\}^\lambda$ is *fresh*, meaning that neither the attacker $\mathcal{A}$ nor the signature scheme has previously made a random oracle call with this value r. If it is not fresh, then the game simply stops and the attacker is defined to lose. Since the attacker and the signature scheme can make at most $\mathsf{poly}(\lambda)$ random oracle calls, the probability of ever choosing a random r which is not fresh is $\mathsf{negl}(\lambda)$. Therefore the games are statistically indistinguishable and $\Pr[E_1] \ge \Pr[E_0] - \mathsf{negl}(\lambda)$.

- Let **Game** 2 be a modified version of **Game** 1 where, in each signing query, instead of the random oracle choosing the values $v_i := H(\tau, r, i)$ uniformly at random (recall, we are guaranteed that r is fresh) and computing $\sigma_i = u_i \leftarrow \mathsf{Inv}_{sk_1, x_i}(v_i)$, we choose a random $\sigma_i = u_i \leftarrow D_{\mathcal{U}}$ and set $v_i = f_{pk_1, x_i}(u_i)$, where x_i is the i'th data symbol of the given signing query. We then program the random oracle $H(\tau, r, i) := v_i$. This change is statistically indistinguishable by the "Distributional Equivalence of Inversion" property of the HTDF. Therefore, $\Pr[E_2] \ge \Pr[E_1] - \mathsf{negl}(\lambda) \ge \Pr[E_0] - \mathsf{negl}(\lambda)$.

- To define **Game** 3, we first define the notion of a *non-homomorphic forgery.* This is a valid forgery $g, \tau, y', (\sigma'_\tau = (r', N', \rho'), \sigma')$ such that the tuple (τ, r', N') does not match any prior signing query: either the label τ was never used in any signing query, or the label $\tau = \tau_j$ was used in a signing query j but $N' \ne N_j$ or $r' \ne r_j$, where N_j was the data-size in query j and r_j was the randomness chosen by the signer in response to query j. Note that non-homomorphic forgeries are a super-set of type I forgeries by also including forgeries with the "wrong" r'. We define **Game** 3 be a modified version of **Game** 2, where the attacker automatically loses if he gives a non-homomorphic forgery. We claim that the security of the standard signature scheme $\mathcal{S}^{nh}$ ensures that $\Pr[E_3] \ge \Pr[E_2] - \mathsf{negl}(\lambda) \ge \Pr[E_0] - \mathsf{negl}(\lambda)$. (*Assume otherwise. Then $\mathcal{A}$ has a non-negligible probability of coming up with a valid non-homomorphic forgery in* **Game** *2. In such a forgery, ρ' is a valid signature of some message (τ, r', N') under the scheme $\mathcal{S}^{nh}$ but this message was never signed under this scheme. Therefore, it is a forgery on $\mathcal{S}^{nh}$.*)

Finally, if the attacker wins in **Game** 3 with some function $g : \mathcal{X}^N \to \mathcal{X}$ and values $\tau, y', (\sigma'_\tau = (r, N, \rho'), \sigma')$, then it must be a type II forgery, where $\tau = \tau_j$, $r = r_j$ and $N = N_j$ match those of some signing query j. We show a reduction that breaks HTDF security whenever this happens. Let $x_1, \ldots, x_N$ be the data chosen by the attacker for signing query j, let u_i and $v_i = f_{pk_1, x_i}(u_i)$ be the values chosen by the challenger when preparing the response to the signing query. Let $y := g(x_1, \ldots, x_N)$, $v^* := \mathsf{Eval}^{out}_{pk_1}(g, v_1, \ldots, v_N)$, $u := \mathsf{Eval}^{in}_{pk_1}(g, (x_1, u_1), \ldots, (x_N, u_N))$. Then, since the signature $u' = \sigma'$ verifies, we have $f_{pk_1, y'}(u') = v^*$. On the other hand, since g is an admissible function, the correctness of homomorphic evaluation ensures that $f_{pk_1, y}(u) = v^*$. Therefore, the values $u, u' \in \mathcal{U}$ and $y \ne y' \in \mathcal{X}$ satisfy $f_{pk', y}(u) = f_{pk', y'}(u')$, and the reduction breaks HTDF security. This shows that $\Pr[E_3] \le \mathsf{negl}(\lambda)$.

Combining all of the above, we see that $\Pr[E_0] \le \Pr[E_3] + \mathsf{negl}(\lambda) \le \mathsf{negl}(\lambda)$ which proves the theorem. □

6 Context-Hiding Security

In many applications, we may also want to guarantee that a signature which certifies y as the output of some computation g over Alice's data should not reveal anything about the underlying data beyond the output of the computation. We will show how to achieve context-hiding by taking our original schemes which produces some signature σ (that is not context hiding) and applying some procedure $\widetilde{\sigma} \leftarrow \mathsf{Hide}_{pk,y}(\sigma)$ which makes the signature context hiding. Once the hiding procedure is applied, the signatures no longer support additional homomorphic operations on them. One additional advantage of this procedure is that it also compresses the size of the signature from $m^2 \log q$ bits to $O(m \log q)$ bits.

Context-Hiding Security for Signatures. We give a simulation-based notion of security, requiring that a context-hiding signature $\widetilde{\sigma}$ can be simulated given knowledge of only the computation g and the output y, but without any other knowledge of Alice's data. The simulation remains indistinguishable even given the underlying data, the underlying signatures, and even the public/secret key of the scheme. In other words, the derived signature does not reveal anything beyond the output of the computation even to an attacker that may have some partial information on the underlying values.

Definition 6.1. *A bounded-data homomorphic signature supports* context hiding *if there exist additional PPT procedures* $\widetilde{\sigma} \leftarrow \mathsf{Hide}_{pk,y}(\sigma)$ *and* $\mathsf{HVerify}_{pk}(g, y, \sigma)$ *such that:*

- *Correctness: For any* $(pk, sk) \in \mathsf{KeyGen}(1^\lambda, 1^N)$ *and any* g, y, σ *such that* $\mathsf{Verify}_{pk}(g, y, \sigma) = \mathtt{accept}$, *for any* $\widetilde{\sigma} \leftarrow \mathsf{Hide}_{pk,y}(\sigma)$ *we have* $\mathsf{HVerify}_{pk}(g, y, \widetilde{\sigma}) = \mathtt{accept}$.
- *Unforgeability: Single-data signature security holds when we replace the* Verify *procedure by* HVerify *in the security game.*
- *Context-Hiding Security: There is a simulator* Sim *such that, for any* fixed *(worst-case) choice of* $(pk, sk) \in \mathsf{KeyGen}(1^\lambda, 1^N)$ *and any* g, y, σ *such that* $\mathsf{Verify}_{pk}(g, y, \sigma) = \mathtt{accept}$ *we have:*

$$\mathsf{Hide}_{pk,y}(\sigma) \approx \mathsf{Sim}(sk, g, y)$$

  where the randomness is only over the random coins of the simulator and the Hide *procedure.*[11] *We say that such schemes are* statistically context hiding *if the above indistinguishability holds statistically.*

The case of multi-data signatures is defined analogously.

Definition 6.2. *A multi-data homomorphic signature supports* context hiding *if there exist additional PPT procedures* $\widetilde{\sigma} \leftarrow \mathsf{Hide}_{pk,x}(\sigma)$, $\mathsf{HVerify}_{pk}(g, y, \tau, (\sigma_\tau, \sigma))$ *such that:*

- *Correctness: For any* $(pk, sk) \in \mathsf{KeyGen}(1^\lambda)$ *and any* $g, y, \sigma_\tau, \sigma$ *such that* $\mathsf{Verify}_{pk}(g, y, \tau, (\sigma_\tau, \sigma)) = \mathtt{accept}$, *for any* $\widetilde{\sigma} \in \mathsf{Hide}_{pk,y}(\sigma)$ *we have* $\mathsf{HVerify}_{pk}(g, y, \tau, (\sigma_\tau, \widetilde{\sigma})) = \mathtt{accept}$.
- *Unforgeability: Multi-data signature security holds when we replace the* Verify *procedure by* HVerify *in the security game.*
- *Context-Hiding Security: Firstly, we require that in the procedure* $(\sigma_\tau, \sigma_1, \ldots, \sigma_N) \in \mathsf{Sign}_{sk}(x_1, \ldots, x_N, \tau)$ *the value* σ_τ *only depends on* (sk, N, τ) *but not on the values* x_i. *Secondly, we require that there is a simulator* Sim *such that, for any* fixed *(worst-case) choice of* $(pk, sk) \in \mathsf{KeyGen}(1^\lambda)$ *and any* $g, y, \sigma, \sigma_\tau$ *such that* $\mathsf{Verify}_{pk}(g, y, \tau, (\sigma_\tau, \sigma)) = \mathtt{accept}$ *we have:*

$$\mathsf{Hide}_{pk,y}(\sigma) \approx \mathsf{Sim}(sk, g, y, \tau, \sigma_\tau)$$

  where the randomness is only over the random coins of the simulator and the Hide *procedure. We say that such schemes are* statistically context hiding *if the above indistinguishability holds statistically.*

[11] Since pk, sk, g, y, σ are fixed, indistinguishability holds even if these values are known to the distinguisher.

Context-Hiding Security for HTDF. We also define a *context hiding HTDF* as an augmentation of standard HTDFs. We will build context-hiding signatures by relying on context-hiding HTDFs.

Definition 6.3. *A context-hiding HTDF comes with two additional algorithms* $\tilde{u} \leftarrow \mathsf{Hide}_{pk,x}(u)$ *and* $\mathsf{Verify}^{HTDF}_{pk}(\tilde{u}, x, v)$ *satisfying:*

- Correctness: *For any* $(pk, sk) \in \mathsf{KeyGen}(1^\lambda)$ *any* $u \in \mathcal{U}$*, any* $x \in \mathcal{X}$ *and any* $\tilde{u} \in \mathsf{Hide}_{pk,x}(u)$ *we have* $\mathsf{Verify}^{HTDF}_{pk}(\tilde{u}, x, f_{pk,x}(u)) = \texttt{accept}$*.*
- Claw-freeness on hidden inputs: *For all PPT* $\mathcal{A}$*:*

$$\Pr\left[\begin{array}{c} \mathsf{Verify}^{HTDF}_{pk}(\tilde{u}', x', f_{pk,x}(u)) = \texttt{accept} \\ u \in \mathcal{U}, x, x' \in \mathcal{X} \quad, x \neq x' \end{array} \;\middle|\; \begin{array}{c} (pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (u, \tilde{u}', x, x') \leftarrow \mathcal{A}(1^\lambda, pk) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

  In other words, if an attacker know u *such that* $f_{pk,x}(u) = v$ *then he cannot also produce* $\tilde{u}'$ *such that* $\mathsf{Verify}^{HTDF}_{pk}(\tilde{u}', x', v) = \texttt{accept}$ *when* $x' \neq x$*.*

- Context Hiding: *There is a simulator* Sim *such that for all choices of* $(pk, sk) \in \mathsf{KeyGen}(1^\lambda), u \in \mathcal{U}$ *and* $x \in \mathcal{X}$ *the following distributions are indistinguishable:*

$$\mathsf{Hide}_{pk,x}(u) \approx \mathsf{Sim}(sk, x, f_{pk,x}(u)).$$

  We say that such schemes are statistically context hiding *if the above indistinguishability holds statistically.*

From Context-Hiding HTDFs to Signatures. We can easily modify the signature schemes constructed in Section 4 (bounded-data) and Section 5 (multi-data) in the natural way to make them context hiding by using a context-hiding HTDF. In particular, the procedure Hide of the signature scheme is defined to be the same as that of the underlying HTDF. The procedure $\mathsf{HVerify}_{pk}(g, y, \widetilde{\sigma})$ of the signature scheme (resp. $\mathsf{HVerify}_{pk}(g, y, \tau, (\sigma_\tau, \widetilde{\sigma}))$ for a multi-data scheme) are defined the same ways as the original Verify procedures of the signature, *except* that, instead of checking $f_{pk,y}(\widetilde{\sigma}) = v^*$ we now check $\mathsf{Verify}^{HTDF}_{pk}(\widetilde{\sigma}, y, v^*) = \texttt{accept}$. It is easy to check that this modification satisfies the given correctness and security requirements as outlined below.

Unforgeability with the modified verification procedure $\mathsf{HVerify}$ follows from the "claw-freeness on hidden inputs" property of the HTDF. This follows from the proofs of Theorem 4.1 and Theorem 5.1. In both proofs, the reduction knows one value $u \in \mathcal{U}$ such that $f_{pk,y}(u) = v^*$ and a signature forgery allows it to come up with $\tilde{u}$ such that $\mathsf{Verify}^{HTDF}_{pk}(\widetilde{\sigma}, y', v^*) = \texttt{accept}$ for $y' \neq y$.

Context-Hiding security of the signature scheme follows from that of the HTDF. We define the signature simulator $\mathsf{Sim}^{sig}(sk, g, y, [\tau, \sigma_\tau])$ to compute the value $v^* = \mathsf{Eval}^{in}_{pk}(g, v_1, \ldots, v_N)$ as is done by the verification procedure of the signature schemes. It then output $\widetilde{\sigma} \leftarrow \mathsf{Sim}^{HTDF}(sk, y, v^*)$. The indistinguishability of the signature simulator follows form that of the HTDF simulator.

General Construction via NIZKs. Before we give our main construction of context-hiding HTDF and therefore context-hiding signatures, we mention that it is possible to solve this problem generically using *non-interactive zero knowledge (ZK) proof of knowledge (PoK) NIZK-PoKs.* In particular, we can make any HTDF context-hiding by setting $\tilde{u} \leftarrow \mathsf{Hide}_{pk,x}(u)$ to be a NIZK-PoK with the statement v and witness u for the relation $f_{pk,x}(u) = v$. The Verify^{HTDF} procedure would simply verify the proof $\tilde{u}$. Claw-freeness follows from the PoK property and context-hiding follows from ZK.[12] However, this approach requires an additional assumption (existence of NIZK-PoK) which is not known to follow from SIS. Therefore, we now proceed to construct context-hiding HTDFs directly.

[12] The syntactic definition would need to be modified slightly to include a common reference string (CRS).

6.1 Construction of Context-Hiding HTDF

Lattice Preliminaries. Before giving our construction of context-hiding HTDFS, we start by recalling some additional useful tools from lattice-based cryptography (abstracting as much as possible). Let $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and let $\mathbf{H} = [\mathbf{A} \mid \mathbf{B}] \in \mathbb{Z}_q^{n \times 2m}$. We will rely on the existence of two algorithms $\mathsf{SampleLeft}$ and $\mathsf{SampleRight}$ which both take $\mathbf{z} \in \mathbb{Z}_q^n$ and manage to output some "short" vector $\mathbf{r} \in \mathbb{Z}_q^{2m}$ such that $\mathbf{H} \cdot \mathbf{r} = \mathbf{z}$. The algorithm $\mathsf{SampleLeft}$ does so by knowing some trapdoor td for the matrix $\mathbf{A}$. The algorithm $\mathsf{SampleRight}$ does so by knowing some "short" matrix $\mathbf{U}$ such that $\mathbf{B} = \mathbf{A}\mathbf{U} + y\mathbf{G}$ for some $y \neq 0$. Nevertheless, the outputs of $\mathsf{SampleLeft}$ and $\mathsf{SampleRight}$ are statistically indistinguishable. (See [CHKP10, ABB10, MP12, BGG$^+$14] for details on the following lemma; our exposition follows [BGG$^+$14] with additional abstraction.)

Lemma 6.4. *Using the notation of Lemma 2.2, let $n, q \geq 2$, $m \geq m^*(n, q)$ and β be parameters. Then there exist polynomial time algorithms $\mathsf{SampleLeft}, \mathsf{SampleRight}$ and some polynomial $p_{extra}(n, m, \log q)$ such that for $\beta' := \beta \cdot p_{extra}(n, m, \log q)$ the following holds: For any choice of $(\mathbf{A}, \mathsf{td}) \in \mathsf{TrapGen}(1^n, 1^m, q)$, any $\mathbf{z} \in \mathbb{Z}_q^n$ and any $\mathbf{U} \in \mathbb{Z}_q^{m \times m}$ with $||\mathbf{U}||_\infty \leq \beta$ and any $y \in \mathbb{Z}_q$ with $y \neq 0$ let $\mathbf{H} = [\mathbf{A} \mid \mathbf{A}\mathbf{U} + y\mathbf{G}]$, where $\mathbf{G}$ is the matrix from part (3) of Lemma 2.2. Then:*

- *For any $\mathbf{r}_0 \in \mathsf{SampleLeft}(\mathbf{H}, \mathsf{td}, \mathbf{z})$, $\mathbf{r}_1 \in \mathsf{SampleRight}(\mathbf{H}, \mathbf{U}, \mathbf{z})$ and for each $b \in \{0, 1\}$ we have $\mathbf{r}_b \in \mathbb{Z}_q^{2m}$, $||\mathbf{r}_b||_\infty \leq \beta'$ and $\mathbf{H} \cdot \mathbf{r}_b = \mathbf{z}$.*
- *For $\mathbf{r}_0 \leftarrow \mathsf{SampleLeft}(\mathbf{H}, \mathsf{td}, \mathbf{z})$ and $\mathbf{r}_1 \leftarrow \mathsf{SampleRight}(\mathbf{H}, \mathbf{U}, \mathbf{z})$ we have $\mathbf{r}_0 \approx \mathbf{r}_1$ are statistically indistinguishable (the statistical distance is negligible in n).*

HTDF with Context Hiding. We augment our construction of HTDFs from Section 3 to add context-hiding security. For this application, we make the following augmentations.

- We restrict the index space $\mathcal{X}$ to just bits $\mathcal{X} = \{0, 1\} \subseteq \mathbb{Z}_q$ (rather than $\mathcal{X} = \mathbb{Z}_q$ as previously). We also modify the parameters and set $\beta_{SIS} = 2^{\omega(\log \lambda)}(\beta_{max})^2$ to be larger than before (which impacts how q, n are chosen to maintain security).
- We augment the public-key to $pk = (\mathbf{A}, \mathbf{z})$ by appending $\mathbf{z} \in \mathbb{Z}_q^n$ which is chosen by selecting a random $r \overset{\$}{\leftarrow} \{0, 1\}^m$ and setting $\mathbf{z} = \mathbf{A} \cdot \mathbf{r}$ (and discarding $\mathbf{r}$).[13] Let $p_{extra}(n, m, \log q) = \mathsf{poly}(\lambda)$ be the polynomial form Lemma 6.4 and define $\tilde{\beta}_{max} = \beta_{max} \cdot p_{extra}(n, m, \log q)$.
- $\tilde{\mathbf{u}} \leftarrow \mathsf{Hide}_{pk,x}(\mathbf{U})$: Let $\mathbf{V} = f_{pk,x}(\mathbf{U}) = \mathbf{A}\mathbf{U} - x\mathbf{G}$. Set

  $$\mathbf{H} := [\mathbf{A} \mid \mathbf{V} + (1 - x)\mathbf{G}] = [\mathbf{A} \mid \mathbf{A}\mathbf{U} + (1 - 2x)\mathbf{G}].$$

  Note that $(1 - 2x) \in \{-1, 1\} \neq 0$. Output $\tilde{\mathbf{u}} \leftarrow \mathsf{SampleRight}(\mathbf{H}, \mathbf{U}, \mathbf{z})$. Note that $\mathbf{H} \cdot \tilde{\mathbf{u}} = \mathbf{z}$ and $||\tilde{\mathbf{u}}||_\infty \leq \tilde{\beta}_{max}$.
- For context-hiding security, we define $\mathsf{Sim}(sk = \mathsf{td}, x, \mathbf{V})$ which computes $\mathbf{H} := [\mathbf{A} \mid \mathbf{V} + (1 - x)\mathbf{G}]$ and outputs $\tilde{\mathbf{u}} \leftarrow \mathsf{SampleLeft}(\mathbf{H}, \mathsf{td}, \mathbf{z})$.
- $\mathsf{Verify}_{pk}^{HTDF}(\tilde{\mathbf{u}}, x, \mathbf{V})$: Compute $\mathbf{H} := [\mathbf{A} \mid \mathbf{V} + (1 - x)\mathbf{G}]$. Check $||\tilde{\mathbf{u}}||_\infty \leq \tilde{\beta}_{max}$ and $\mathbf{H} \cdot \tilde{\mathbf{u}} = \mathbf{z}$. If so accept, else reject.

It's easy to check that correctness holds. Context hiding security follows directly from Lemma 6.4. We are left to show claw-freeness on hidden inputs.

[13] We note that, using the leftover-hash-lemma, we can show that this is statistically close to choosing $\mathbf{z} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$ at random. However, we will not need to rely on this fact.

Theorem 6.5. *The above scheme satisfied* claw-freeness on hidden inputs *under the* $\mathsf{SIS}(n, m, q, \beta_{SIS})$ *assumption.*

Proof. The proof of security closely follows that of Theorem 3.1. Assume that $\mathcal{A}$ is a PPT attacker that breaks this security property of the scheme. As a first step, we modify the game so that, instead of sampling $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ and setting $pk := \mathbf{A}$ and $sk = \mathsf{td}$, we just choose $\mathbf{A} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n \times m}$ uniformly at random. This modification is statistically indistinguishable by the security of $\mathsf{TrapGen}$ (see Lemma 2.2, part (2)). In particular, the probability of $\mathcal{A}$ winning the modified game remains non-negligible.

We now show that an attacker who wins the above-modified game can be used to solve the SIS problem. The reduction gets a challenge matrix $\mathbf{A}$ of the SIS problem and chooses $\mathbf{r} \overset{\$}{\leftarrow} \{0,1\}^m$ and sets $\mathbf{z} = \mathbf{A} \cdot \mathbf{r}$. It gives the public key $pk = (\mathbf{A}, \mathbf{z})$ to the attacker $\mathcal{A}$. The attacker wins if he comes up with bits $x \neq x' \in \{0,1\}$ and values $\mathbf{U}, \tilde{\mathbf{u}}'$ such that $||\mathbf{U}||_\infty \leq \beta_{max}$, $||\tilde{\mathbf{u}}'||_\infty \leq \tilde{\beta}_{max}$, and $\mathbf{H} \cdot \tilde{\mathbf{u}}' = \mathbf{z}$ where $\mathbf{H}$ is defined by setting $\mathbf{V} := f_{pk,x}(\mathbf{U}) = \mathbf{AU} - x\mathbf{G}$ and

$$\mathbf{H} := [\mathbf{A} \mid \mathbf{V} + (1 - x')\mathbf{G}] = [\mathbf{A} \mid \mathbf{AU} + (1 - (x + x'))\mathbf{G}] = [\mathbf{A} \mid \mathbf{AU}]$$

where the last equality follows since $x \neq x' \;\Rightarrow\; (x + x') = 1$. Let's write $\tilde{\mathbf{u}}' = (\mathbf{r}_1', \mathbf{r}_2')$ where $\mathbf{r}_1', \mathbf{r}_2' \in \mathbb{Z}_q^m$ are the first and last m components of $\tilde{\mathbf{u}}'$ respectively. Then:

$$\mathbf{H} \cdot \tilde{\mathbf{u}}' = \mathbf{z} \quad\Rightarrow\quad \mathbf{A}\mathbf{r}_1 + (\mathbf{AU})\mathbf{r}_2 = \mathbf{Ar} \quad\Rightarrow\quad \mathbf{A}(\mathbf{U}\mathbf{r}_2 + \mathbf{r}_1 - \mathbf{r}) = \mathbf{0}$$

Furthermore

$$||(\mathbf{U}\mathbf{r}_2 + \mathbf{r}_1 - \mathbf{r})||_\infty \leq m\beta_{max}\tilde{\beta}_{max} + \tilde{\beta}_{max} + 1 \leq \mathsf{poly}(\lambda)(\beta_{max})^2 \leq \beta_{SIS}.$$

Therefore, it remains to show that $(\mathbf{U}\mathbf{r}_2 + \mathbf{r}_1 - \mathbf{r}) \neq \mathbf{0}$. We use the same argument as in the proof of Theorem 3.1: the randomness $\mathbf{r}$ is independent of $\mathbf{U}, \mathbf{r}_1, \mathbf{r}_2$ when conditioned on $\mathbf{z}$. Since $\mathbf{z}$ is short, $\mathbf{r}$ still has $m - n \log q = \omega(\log \lambda)$ bits of conditional entropy left and therefore $\Pr[\mathbf{U}\mathbf{r}_2 + \mathbf{r}_1 = \mathbf{r}] \leq \mathsf{negl}(\lambda)$. This concludes the proof. □

7 Conclusions

In this work, we construct the first leveled fully homomorphic signature schemes. It remains an open problem to get rid of the *leveled* aspect and ideally come up with a signature scheme where there is no a priori bound on the depth of the circuits that can be evaluated and the signature size stays fixed. Another open problem is to design multi-data homomorphic signature schemes in the standard model.

8 Acknowledgements

We thank Valeria Nikolaenko for patiently explaining her result with Boneh et al. [BGG$^+$14] to us. We rely on the core technical idea of that result to perform homomorphic multiplications in this work.

References

[AB09] Shweta Agrawal and Dan Boneh. Homomorphic MACs: MAC-based integrity for network coding. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS 09*, volume 5536 of *LNCS*, pages 292–305, Paris-Rocquencourt, France, June 2–5, 2009. Springer, Berlin, Germany.

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h)ibe in the standard model. In Gilbert [Gil10], pages 553–572.

[ABC$^+$07] Giuseppe Ateniese, Randal C. Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary N. J. Peterson, and Dawn Song. Provable data possession at untrusted stores. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 07*, pages 598–609, Alexandria, Virginia, USA, October 28–31, 2007. ACM Press.

[ABC$^+$12] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, Abhi Shelat, and Brent Waters. Computing on authenticated data. In Ronald Cramer, editor, *TCC*, volume 7194 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2012.

[AIK10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *ICALP 2010, Part I*, volume 6198 of *LNCS*, pages 152–163, Bordeaux, France, July 6–10, 2010. Springer, Berlin, Germany.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In Gary L. Miller, editor, *STOC*, pages 99–108. ACM, 1996.

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP*, volume 1644 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 1999.

[AKK09] Giuseppe Ateniese, Seny Kamara, and Jonathan Katz. Proofs of storage from homomorphic identification protocols. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 319–333, Tokyo, Japan, December 6–10, 2009. Springer, Berlin, Germany.

[AL11] Nuttapong Attrapadung and Benoît Libert. Homomorphic network coding signatures in the standard model. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 17–34, Taormina, Italy, March 6–9, 2011. Springer, Berlin, Germany.

[AP09] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In Susanne Albers and Jean-Yves Marion, editors, *STACS*, volume 3 of *LIPIcs*, pages 75–86. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *ITCS*, pages 326–349. ACM, 2012.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *STOC*, pages 111–120. ACM, 2013.

[BCI$^+$13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, pages 315–333, 2013.

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. *Proceedings of the 46th annual ACM symposium on Symposium on theory of computing*, 2014.

[BF11a] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 149–168, Tallinn, Estonia, May 15–19, 2011. Springer, Berlin, Germany.

[BF11b] Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 1–16, Taormina, Italy, March 6–9, 2011. Springer, Berlin, Germany.

[BFKW09] Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 68–87, Irvine, CA, USA, March 18–20, 2009. Springer, Berlin, Germany.

[BFR13] Michael Backes, Dario Fiore, and Raphael M. Reischuk. Verifiable delegation of computation on outsourced data. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM Conference on Computer and Communications Security*, pages 863–874. ACM, 2013.

[BGG$^+$14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT*, volume 8441 of *Lecture Notes in Computer Science*, pages 533–556. Springer, 2014.

[BGV11] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. Verifiable delegation of computation over large datasets. In Rogaway [Rog11], pages 111–131.

[BSCG$^+$13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO*, volume 8043 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2013.

[CF13] Dario Catalano and Dario Fiore. Practical homomorphic macs for arithmetic circuits. In Johansson and Nguyen [JN13], pages 336–352.

[CFGN14] Dario Catalano, Dario Fiore, Rosario Gennaro, and Luca Nizzardo. Generalizing homomorphic macs for arithmetic circuits. In Hugo Krawczyk, editor, *Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 538–555. Springer, 2014.

[CFW12] Dario Catalano, Dario Fiore, and Bogdan Warinschi. Efficient network coding signatures in the standard model. In Fischlin et al. [FBM12], pages 680–696.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Gilbert [Gil10], pages 523–552.

[CKLR11] Kai-Min Chung, Yael Tauman Kalai, Feng-Hao Liu, and Ran Raz. Memory delegation. In Rogaway [Rog11], pages 151–168.

[CKV10] Kai-Min Chung, Yael Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 483–501, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Berlin, Germany.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[DVW09] Yevgeniy Dodis, Salil P. Vadhan, and Daniel Wichs. Proofs of retrievability via hardness amplification. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 109–127. Springer, Berlin, Germany, March 15–17, 2009.

[FBM12] Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors. *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*, volume 7293 of *Lecture Notes in Computer Science*. Springer, 2012.

[Fre12] David Mandell Freeman. Improved security for linearly homomorphic signatures: A generic framework. In Fischlin et al. [FBM12], pages 697–714.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178, Bethesda, Maryland, USA, May 31 – June 2, 2009. ACM Press.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Berlin, Germany.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In Johansson and Nguyen [JN13], pages 626–645.

[Gil10] Henri Gilbert, editor. *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*. Springer, 2010.

[GKKR10] Rosario Gennaro, Jonathan Katz, Hugo Krawczyk, and Tal Rabin. Secure network coding over the integers. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 142–160, Paris, France, May 26–28, 2010. Springer, Berlin, Germany.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 113–122, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *STOC*, pages 197–206. ACM, 2008.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108, San Jose, California, USA, June 6–8, 2011. ACM Press.

[GW13] Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT (2)*, volume 8270 of *Lecture Notes in Computer Science*, pages 301–320. Springer, 2013.

[JMSW02] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In Bart Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 244–262, San Jose, CA, USA, February 18–22, 2002. Springer, Berlin, Germany.

[JN13] Thomas Johansson and Phong Q. Nguyen, editors. *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*. Springer, 2013.

[KRR14] Yael Tauman Kalai, Ran Raz, and Ron Rothblum. How to delegate computations: The power of no-signaling proofs. *Proceedings of the 46th annual ACM symposium on Symposium on theory of computing*, 2014.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *FOCS*, pages 436–453. IEEE Computer Society, 1994.

[Mic04] Daniele Micciancio. Almost perfect lattices, the covering radius problem, and applications to ajtai's connection factor. *SIAM J. Comput.*, 34(1):118–169, 2004.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, 2012.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of sis and lwe with small parameters. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2013.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society, 2013.

[PST13] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In *TCC*, pages 222–242, 2013.

[Rog11] Phillip Rogaway, editor. *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*. Springer, 2011.

[SW08] Hovav Shacham and Brent Waters. Compact proofs of retrievability. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 90–107, Melbourne, Australia, December 7–11, 2008. Springer, Berlin, Germany.