Leveled Fully Homomorphic Signatures from Standard Lattices

Sergey Gorbunov[*] MIT

Vinod Vaikuntanathan[†] MIT

Daniel Wichs[‡] Northeastern

Abstract

In a homomorphic signature scheme, a user Alice signs some large dataset x using her secret signing key and uploads the signed data to an untrusted remote server. The server can then run some computation $y = f(x)$ over the signed data and homomorphically derive a *short* signature $\sigma_{f,y}$ certifying that y is the correct output of the computation f. Anybody can verify the tuple $(f, y, \sigma_{f,y})$ using Alice's public verification key and become convinced of this fact without having to retrieve the entire underlying data.

In this work, we construct the first *(leveled) fully homomorphic signature* schemes that can evaluate arbitrary circuits over signed data. Only the maximal depth d of the circuits needs to be fixed a-priori at setup, and the size of the evaluated signature grows polynomially in d, but is otherwise independent of the circuit size or the data size. Our solution is based on the hardness of the *small integer solution* (SIS) problem in standard lattices and satisfies full (adaptive) security. In the standard model, we get a scheme with large public parameters whose size exceeds the total size of a dataset. In the random-oracle model, we get a scheme with short public parameters. In both cases, the schemes can be used to sign many different datasets. The complexity of verifying a signature for a computation f is at least as large as that of computing f, but can be amortized when verifying the same computation over many different datasets. Furthermore, the signatures can be made *context-hiding* so as not to reveal anything about the data beyond the outcome of the computation.

These results offer a significant improvement in capabilities and assumptions over the best prior homomorphic signature schemes, which were limited to evaluating polynomials of constant degree.

As a building block of independent interest, we introduce a new notion called *homomorphic trapdoor functions* (HTDF) which conceptually unites homomorphic encryption and signatures. We construct HTDFs by relying on the techniques developed by Gentry et al. (CRYPTO '13) and Boneh et al. (EUROCRYPT '14) in the contexts of fully homomorphic and attribute-based encryptions.

[*]Email: `sergeyg@mit.edu`. Research supported in part by Microsoft PhD Fellowship and the Northrop Grumman Cybersecurity Research Consortium (CRC).

[†]Email: `vinodv@csail.mit.edu`. Research supported in part by DARPA Grant number FA8750-11-2-0225, the Northrop Grumman Cybersecurity Research Consortium (CRC), Alfred P. Sloan Research Fellowship, Microsoft Faculty Fellowship, NSF CAREER award CNS-1350619, NSF Frontier Grant CNS-1414119, and the Steven and Renee Finn Career Development Chair from MIT.

[‡]E-mail: `wichs@ccs.neu.edu`. Research supported by NSF grants CNS-1347350 and CNS-1314722.

Contents

1 Introduction **1**
1.1 Our Results . . . 2
1.2 Related Work . . . 3
1.3 Our Techniques . . . 4

2 Preliminaries **7**
2.1 Background on Lattices and the SIS Problem . . . 8

3 Homomorphic Trapdoor Functions **9**
3.1 Definition . . . 9
3.2 Construction: Basic Algorithms and Security . . . 10
3.3 Construction: Homomorphic Evaluation and Noise Growth . . . 12

4 Fully Homomorphic Signatures (Single Dataset) **14**
4.1 Definition . . . 15
4.2 Basic Construction . . . 17
4.3 A Scheme with Short Public Parameters . . . 18
4.4 From Selective Security to Full Security . . . 19

5 Multi-Data Homomorphic Signatures **21**
5.1 Definition . . . 21
5.2 From Single-Data to Multi-Data . . . 23

6 Context-Hiding Security **25**
6.1 Construction of Context-Hiding HTDF . . . 27

7 Conclusions **29**

A Another Single to Multi-Data Transformation **34**

B HTDFs and Fully Homomorphic Encryption **36**

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 sign some large dataset x using her secret signing key. She can then distribute the signed data to some untrusted entity called a "data processor". A data processor can perform arbitrary computations $y = f(x)$ over this data and homomorphically derive a signature $\sigma_{f,y}$, which certifies that y is the correct output of the computation f over Alice's data. The derived signature $\sigma_{f,y}$ should be short, with length independent of the size of the data x. Anybody can verify the tuple $(f, y, \sigma_{f,y})$ using Alice's public verification key and become convinced that y is indeed the correct output of the computation f over Alice's signed data x, without needing to download the entire data x. Although the computational effort of verifying a derived signature is proportional to the complexity of computing the function f, this work can be performed *offline* prior to seeing the signature, and can be amortized when verifying the same computation over many datasets.

Application: Computation on Outsourced Data. As the most basic application of homomorphic signatures, a user Alice can outsource her signed data to a remote server acting as a data processor, and later verify various computations performed by the server over her data. Using homomorphic signatures, this can be done with *minimal communication and interaction* consisting of a single short signature sent from the server to Alice. Although verifying the correctness of this computation takes Alice as much time as the computation itself, she avoids having to store this data long term. We refer the reader to Section 1.2 for a detailed comparison with related work on delegating memory and computation.

Application: Certified Data Analysis. The *non-interactive* nature of homomorphic signatures and the fact that they provide *public verifiability* also makes them useful in a wide variety of settings beyond the above outsourcing scenario.

For example, consider a scenario where a trusted agency such as the National Institute of Health (NIH), runs a large-scale medical study. It signs the collected data x and distributes it to various research groups and pharmaceutical companies for analysis. Some of these groups may have incentives to lie about the outcomes of their analysis and are not trusted by the public. However, using homomorphic signatures, they can publicly post their methodology for the analysis (a function f), the claimed outcome of the analysis ($y = f(x)$), and a short signature $\sigma_{f,y}$ that certifies the correctness of the outcome. This information can be posted publicly (e.g., on third-party websites) and verified *by the public* using a verification key published by the NIH. The public *does not need to have access to the underlying data* and *does not need to interact* with the NIH or the research groups that performed the computation to verify such signatures – indeed, these entities may go offline and the underlying data may be deleted after the analysis is performed but the signed results remain verifiable. Furthermore, such signatures can be made *context hiding* to ensure that they do not reveal anything about the underlying medical data beyond the outcome of the analysis. In this case, the NIH trusts the research groups to preserve the privacy of the underlying data (from the world), but the world does not trust the research groups to perform the analysis correctly.

1.1 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 data size or the circuit size. This is an exponential improvement in capabilities over the best prior homomorphic signature schemes which could only evaluate polynomials of some bounded degree k where the efficiency degraded polynomially with k (in general, $k = 2^{O(d)}$).

Our solutions are based on the (subexponential) hardness of the *small integer solution* (SIS) problem in standard lattices, which is in turn implied by the worst-case hardness of standard lattice problems [Ajt96]. This is also a significant improvement in assumptions over the best prior schemes which either relied on ideal lattices or multi-linear maps.

We get a scheme in the standard model, where the maximal dataset size needs to be bounded by some polynomial N during setup and the size of the public parameters is linear in N. In the random-oracle model, we get rid of this caveat and get a scheme with short public parameters and without any a-priori bound on the dataset size. In both cases, the user can sign arbitrarily many different datasets by associating each dataset with some label (e.g., a file name). The verifier must know the label of the dataset on which the computation was supposed to be performed when verifying the output.

Efficient Online/Amortized Verification. Our schemes allow for fast amortized verification of a computation f over many different datasets, even if the datasets belong to different users with different verification keys. In particular, the verifier only needs to perform work proportional to the circuit size of f once to verify arbitrarily many signatures under arbitrary verification keys. This work can be performed offline prior to receiving any of the signatures, and the online verification can then be much more efficient than computing f.

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

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 outputs 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 $(g \circ \bar{h})(x) \overset{\text{def}}{=} g(h_1(x), \ldots, h_\ell(x))$. 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.

1.2 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 and 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. The recent work of Catalano et al. [CFW14] gives an alternate solution using multi-linear maps which removes the need for random oracles at the cost of having large public parameters. The main open question left by these works is to construct 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)* with 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 authentication tag stays fixed). In other words, the question of (leveled) fully homomorphic authentication, even in the setting of private verification, remained open prior to this work.

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 general notion of P-homomorphic signature schemes for various predicates P, but efficient constructions were only known for a few specific instances. [1]

[1]There is a potentially confusing syntactic difference between the notion of P-homomorphic signatures and the notion of homomorphic signatures in this work, although the two are equivalent. In P-homomorphic signatures, given a signature of x one should be able to derive a signature of x' as long as $P(x, x') = 1$ for some predicate P (e.g., the substring predicate). The same effect can be achieved using the syntax of homomorphic signatures in this work by defining a function $f_{x'}$ that has x' hard-coded and computes $f_{x'}(x) = P(x, x')$. We would then give a derived signature $\sigma_{f_{x'},1}$ certifying that $y = 1$ is the output of the computation $f_{x'}(x)$ over the originally signed data x.

Homomorphic Signatures via SNARKs. There is a very simple construction of fully homomorphic signatures by relying on 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 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 $y = f(x)$ of any computation f over any signed data x (under any standard signature scheme) by creating a short argument $\pi_{f,y}$ that proves the knowledge of "data x along with valid signature of x, such that $f(x) = y$".

One advantage of the SNARK-based scheme is that a signature can be verified very efficiently, independently of the complexity of the computation f being verified, as long as f has a short Turing-Machine description. In contrast, in this work, we will only get efficient verification in an amortized sense, when verifying a computation f over many different datasets. 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 PCP machinery, or on various "knowledge of exponent" assumptions and light-weight PCP variants.

Delegating Computation. 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, PRV12, PST13, KRR14]. In this scenario, the server needs to convince the user that $f(x) = y$, where the user knows the function f, the input x and the output y, but does not want to do the *work* of computing $f(x)$. In contrast, in our scenario the verifier only knows f and y but does *not* know the previously authenticated data x, which may be huge. As mentioned in [GW13], some of the results for delegating computation in the pre-processing model can also be re-interpreted as giving results for the latter scenario. The latter scenario was also explicitly considered by Chung et al. [CKLR11] in the context of *memory delegation*, where the client can also update the data on the server. Some of these solutions only allow a single party with secret key to verify computation, while others (e.g., [PRV12]) allow anyone to verify. However, all of the above solutions for memory delegation and delegating computation require at least some interaction between the client and server (often just a challenge-response protocol) to verify a computation f. Therefore they do not give us a solution to the problem of homomorphic signatures (or even homomorphic MACs), where we require a *static* certificate which certifies the output of a computation and which can be posted publicly and verified by everyone.

1.3 Our Techniques

Our constructions of homomorphic signatures are modular and, as a building block of potentially independent interest, we present a new primitive called a *homomorphic trapdoor function* (HTDF). This primitive allows us to conceptually unite homomorphic encryption and signatures. 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 $v = f_{pk,x}(u)$ described via a public key pk and an index $x \in \{0,1\}$, with input u and output v. It will be useful to think of this as a commitment scheme where x is the message, v is the commitment and u is the randomness/decommitment. Given some values

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

and a circuit $g : \{0,1\}^N \to \{0,1\}$, we can homomorphically compute an input $u^* := \mathsf{HTDF.Eval}^{in}(g, (x_1, u_1) \ldots, (x_N, u_N))$ and an output $v^* := \mathsf{HTDF.Eval}^{out}(g, v_1, \ldots, v_N)$ such that:

$$f_{pk,g(x_1,\ldots,x_N)}(u^*) = v^*.$$

Thinking of HTDFs as commitments, the above says that if the values v_i are commitments to the message bits x_i with decommitments u_i, then we can homomorphically combine the v_i to derive a commitment v^* and homomorphically combine the messages/decommitments (x_i, u_i) to get a decommitment u^* which opens v^* to the message $g(x_1, \ldots, x_N)$.

We want to be able to generate the public key pk of the HTDF together with a *trapdoor sk* that allows us to take any output v and efficiently invert it with respect to any index x to get $u \leftarrow \mathsf{Inv}_{sk,x}(v)$ such that $f_{pk,x}(u) = v$. In the language of commitments, this means that we want the scheme to be equivocable (and therefore statistically hiding) with a trapdoor sk that lets us open any commitment to any message.

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 $f_{pk,0}(u_0) = f_{pk,1}(u_1)$. Equivalently, in the language of commitments, we want the scheme to be computationally binding.

As an intellectual curiosity (but of no application to this work), we could also change our requirements and instead ask for an HTDF which is extractable (and therefore statistically binding) while also being computationally hiding. In other words, we would want to generate pk along with a trapdoor sk that would allow us to extract x from $v = f_{pk,x}(u)$. In this case, such an HTDF could also be though of as a fully homomorphic encryption scheme, where v is the ciphertext of a message x and $\mathsf{HTDF.Eval}^{out}$ is the homomorphic evaluation procedure on ciphertexts. Our eventual construction of an HTDF will provide both options by allowing us to choose pk in one of two indistinguishable modes: an equivocable mode and an extractable mode. In this work, we will solely rely on the equivocable mode to construct homomorphic signatures. However, the extractable mode of the HTDF (essentially) corresponds to the Gentry-Sahai-Waters [GSW13] fully homomorphic encryption scheme. Therefore, we view HTDFs as providing an interesting conceptual unification of homomorphic signatures and encryption. We refer the reader to Appendix B for more on this grand unification.

Basic Homomorphic Signatures from HTDFs. We construct several flavors of homomorphic signature schemes using HTDFs as a black-box. As the most basic flavor, we construct a signature scheme in the standard model where the setup procedure knows some bound N on the size of the dataset 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 $x = (x_1, \ldots, x_N) \in \{0,1\}^N$ 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 u_i as a signature that ties x_i to its position i.

Given x, the signatures $u_1, \ldots, u_N$, and a function $g : \{0,1\}^N \rightarrow \{0,1\}$, anybody can homomorphically compute a signature $u^*_{g,y} := \mathsf{HTDF.Eval}^{in}(g, (x_1, u_1), \ldots, (x_N, u_N))$ 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 will compute $v^* := \mathsf{HTDF.Eval}^{out}(g, v_1, \ldots, v_N)$ and checks $f_{pk,y}(u^*_{g,y}) \overset{?}{=} v^*$. Notice that verification procedure only depends on the public parameters but does not know the data x. We can show that this basic scheme already satisfies selective security, where we assume that dataset $x_1, \ldots, x_N$ 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 compute $v_i := f_{pk,x_i}(u_i)$ using the data x_i that the attacker wants signed. This makes it easy for the reduction to generate signatures for x_i. Furthermore, for any function g, the reduction can compute the honest signature $u = \mathsf{HTDF.Eval}^{in}(g, (x_1, u_1), \ldots, (x_N, u_N))$ which certifies the output $y = g(x_1, \ldots, x_N)$. If an attacker produces a forged signature u' that certifies $y' \neq y$ then $f_{pk,y}(u) = f_{pk,y'}(u')$ and therefore (u, u') breaks the claw-free security of the HTDF.

Upgrading Functionality and Security. We show how to generically start with a basic homomorphic signature scheme as above and convert into more powerful variants of homomorphic signatures with improved functionality, efficiency, and security.

Firstly, we note that since the public parameters $\mathsf{prms} = (v_1, \ldots, v_N)$ of our basic scheme are uniformly random values, we can easily compress them in the random oracle model to get a scheme with short public parameters. In particular, the public parameters of the new scheme only consist of a short random string r and we can derive the values $v_i = H(r, i)$ using a random oracle H. We can also translate this random-oracle scheme into a standard-model assumption on the hash function H which is simple-to-state and falsifiable, but nevertheless non-standard. This gives us a tradeoff between efficiency and assumptions.

Next, we give a generic transformation from a homomorphic signature scheme with selective security to a scheme with full adaptive security. Our transformation works in both the standard model and the random oracle model. Starting from a selectively secure leveled FHS scheme, we obtain a fully secure leveled FHS scheme.[2]

Lastly, following Boneh and Freeman [BF11a], we can extend the functionality of homomorphic signatures to allow the user to sign multiple different datasets under different *labels* τ (e.g., τ can correspond to a "file name"), where verifier must simply know the label of the dataset on which the computation was supposed to be performed. We show a generic transformation from a basic signature that only works for a single dataset into one that supports multiple datasets. Furthermore, this transformation gives us efficient amortized verification of a computation over multiple datasets.

Constructing HTDFs. We now briefly describe how to construct HTDFs based on the SIS problem. We rely on the homomorphic techniques developed by Gentry, Sahai and Waters [GSW13] and by Boneh et al. [BGG+14] in the context of fully homomorphic encryption and attribute-based encryption.

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 $\mathbf{V} \in \mathbb{Z}_q^{n \times m}$,

[2]Although Catalano et al. [CFW14] provide a similar transformation, it works only for bounded degree polynomial functions, and does not generalize to leveled FHS.

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 can efficiently compute a "short" matrix $\mathbf{G}^{-1}(\mathbf{V})$ such that $\mathbf{G}\mathbf{G}^{-1}(\mathbf{V}) = \mathbf{V}$.[3]

Our HTDF consists of choosing $pk = \mathbf{A}$ together with trapdoor $sk = \mathsf{td}$ as above. We define $f_{pk,x}(\mathbf{U}) \stackrel{\text{def}}{=} \mathbf{AU} + x \cdot \mathbf{G}$, but we restrict the domain to "short" matrices $\mathbf{U}$. We show that finding a claw consisting of "short" matrices $\mathbf{U}_0, \mathbf{U}_1$ such that $f_{pk,0}(\mathbf{U}_0) = f_{pk,1}(\mathbf{U}_1) \Rightarrow \mathbf{A}(\mathbf{U}_0 - \mathbf{U}_1) = \mathbf{G}$ implies breaking the SIS problem. 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

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

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 $\mathbf{U}^* = \mathbf{U}_1 + \mathbf{U}_2$. This ensures:

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

Multiplication. Homomorphic multiplication (over $\mathbb{Z}_q$) consists of setting $\mathbf{V}^* := \mathbf{V}_2\mathbf{G}^{-1}(\mathbf{V}_1)$ and $\mathbf{U}^* := x_2\mathbf{U}_1 + \mathbf{U}_2\mathbf{G}^{-1}(\mathbf{V}_1)$. This gives:

$$\begin{aligned} \mathbf{V}^* &= \mathbf{V}_2\mathbf{G}^{-1}(\mathbf{V}_1) = (\mathbf{AU}_2 + x_2\mathbf{G})\mathbf{G}^{-1}(\mathbf{V}_1) = \mathbf{AU}_2\mathbf{G}^{-1}(\mathbf{V}_1) + x_2(\mathbf{AU}_1 + x_1\mathbf{G}) \\ &= \mathbf{AU}^* + x_1x_2\mathbf{G} = f_{pk,x_1 \cdot x_2}(\mathbf{U}^*) \end{aligned}$$

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 to at most $(m+1)\beta$. Therefore, when evaluating a boolean circuit of depth d the noise level can grow to as much as $\beta(m+1)^d$. We pause to note that the noise growth 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, meaning that evaluating a circuit of depth d could raise the noise level to as high as β^{2^d}. Since the modulus must satisfy $q \gg \beta(m+1)^d$ for security, the level of homomorphism d must be 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 $[N] \stackrel{\text{def}}{=} \{1, \ldots, N\}$. 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 $x \stackrel{\$}{\leftarrow} \mathcal{X}$ 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

[3] Note that we are abusing notation and $\mathbf{G}^{-1}$ is not a matrix but rather a function - for any $\mathbf{V}$ there are many choices of $\mathbf{U}$ such that $\mathbf{GU} = \mathbf{V}$, and $\mathbf{G}^{-1}(\mathbf{V})$ deterministically outputs a particular short matrix from this set. For those familiar with [GSW13], multiplication by $\mathbf{G}$ corresponds to $\mathsf{PowersOf2}()$ and $\mathbf{G}^{-1}()$ corresponds to $\mathsf{BitDecomp}()$.

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 $||\mathbf{u}||_\infty \leq \beta$ such that $\mathbf{A} \cdot \mathbf{u} = \mathbf{0}$.[4] 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

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

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.

[4]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.

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* $n \geq 1, q \geq 2$ *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* $m \geq m^*$ *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* $||\mathbf{U}||_\infty \leq \beta_{sam}$ *(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* $||\mathbf{U}'||_\infty \leq \beta_{sam}$.

3. *Given* n, m, q *as above, there is an efficiently and deterministically computable matrix* $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ *and a deterministic polynomial-time algorithm* $\mathbf{G}^{-1}$ *which takes the input* $\mathbf{V} \in \mathbb{Z}_q^{n \times k}$ *for any integer* k *and outputs* $\mathbf{R} = \mathbf{G}^{-1}(\mathbf{V})$ *such that* $\mathbf{R} \in \{0,1\}^{m \times k}$ *and* $\mathbf{G} \cdot \mathbf{R} = \mathbf{V}$*.* [5]

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 ($\mathsf{HTDF.KeyGen}$, f, Inv, $\mathsf{HTDF.Eval}^{in}$, $\mathsf{HTDF.Eval}^{out}$) with syntax:

- $(pk, sk) \leftarrow \mathsf{HTDF.KeyGen}(1^\lambda)$: 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.

[5] Note that we are abusing notation and $\mathbf{G}^{-1}$ is not a matrix but rather an algorithm. See footnote 3.

- $u^* = \mathsf{HTDF.Eval}^{in}(g, (x_1, u_1), \ldots, (x_\ell, u_\ell))$, $v^* = \mathsf{HTDF.Eval}^{out}(g, v_1, \ldots, v_\ell)$ 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 $v^* \in \mathcal{V}$.[6]

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 $(pk, sk) \in \mathsf{HTDF.KeyGen}(1^\lambda)$,[7] $x_1, \ldots, x_\ell \in \mathcal{X}$, $g : \mathcal{X}^\ell \to \mathcal{X}$ 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 $i \in [\ell]$. Let $u^* := \mathsf{HTDF.Eval}^{in}(g, (x_1, u_1), \ldots, (x_\ell, u_\ell))$, $v^* := \mathsf{HTDF.Eval}^{out}(g, v_1, \ldots, v_\ell)$. Then we require that $u^* \in \mathcal{U}$ and $f_{pk,y}(u^*) = v^*$.

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 $\beta_i \le \beta_{init}$, then $u^* := \mathsf{HTDF.Eval}^{in}(g, (x_1, u_1), \ldots, (x_\ell, u_\ell))$ will have noise-level $\beta^* \le \beta_{max}$.

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

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:

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

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.

[6]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.

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

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.

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

- Define the domains $\mathcal{X} = \mathbb{Z}_q$ and $\mathcal{V} = \mathbb{Z}_q^{n\times m}$. Let $\mathcal{U} = \{\mathbf{U} \in \mathbb{Z}_q^{m\times m} \ : \ ||\mathbf{U}||_\infty \le \beta_{max}\}$. 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 $||\mathbf{U}||_\infty \le \beta_{init}$.
- $(pk, sk) \leftarrow \mathsf{HTDF.KeyGen}(1^\lambda)$: 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 $(pk = \mathbf{A}, sk = \mathsf{td}) \leftarrow \mathsf{HTDF.KeyGen}(1^\lambda)$, and let $x \in \mathcal{X}$ be an arbitrary random variable that depends on (pk, sk). Let $\mathbf{U} \leftarrow D_{\mathcal{U}}$, $\mathbf{V} = \mathbf{A}\mathbf{U}+x\cdot\mathbf{G} = f_{pk,x}(\mathbf{U})$, $\mathbf{V}' \overset{\$}{\leftarrow} \mathcal{V}$, $\mathbf{U}' \leftarrow \{\mathsf{Inv}_{sk,x}(\mathbf{V}') = \mathsf{SamPre}(\mathbf{A}, \mathbf{V}' - x \cdot \mathbf{G}, \mathsf{td})\}$. Then we need to show:

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

Lemma 2.2, part (2) tells us that:

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

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 $\mathbf{r}' = \mathbf{G}^{-1}(\mathbf{z}/x^*)$ 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 $\mathsf{HTDF.Eval}^{in}$, $\mathsf{HTDF.Eval}^{out}$ with the syntax

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

Our approach closely follows the techniques of [GSW13, 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 $\mathsf{HTDF.Eval}^{in}, \mathsf{HTDF.Eval}^{out}$ 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 $\mathsf{HTDF.Eval}^{in}$ ignores the values x_1, x_2.

- Let $g(x_1, x_2) = x_1 \cdot x_2$ be a *multiplication* gate. Let $\mathbf{R} = \mathbf{G}^{-1}(\mathbf{V}_1)$ so that $\mathbf{R} \in \{0,1\}^{m \times m}$ and $\mathbf{GR} = -\mathbf{V}_1$. The algorithms $\mathsf{HTDF.Eval}^{in}, \mathsf{HTDF.Eval}^{out}$ respectively compute:
$$\mathbf{U}^* := x_2 \cdot \mathbf{U}_1 + \mathbf{U}_2 \mathbf{G}^{-1}(\mathbf{V}_1) \quad , \quad \mathbf{V}^* := \mathbf{V}_2 \cdot \mathbf{G}^{-1}(\mathbf{V}_1).$$
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 $\mathsf{HTDF.Eval}^{in}, \mathsf{HTDF.Eval}^{out}$ 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 $\mathsf{HTDF.Eval}^{in}, \mathsf{HTDF.Eval}^{out}$ 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, the algorithms $\mathsf{HTDF.Eval}^{in}, \mathsf{HTDF.Eval}^{out}$ respectively compute:
$$\mathbf{U}^* := \mathbf{U} \cdot \mathbf{G}^{-1}(a \cdot \mathbf{G}) \quad , \quad \mathbf{V}^* := \mathbf{V} \cdot \mathbf{G}^{-1}(a \cdot \mathbf{G}).$$
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 $\mathsf{NAND}(x_1, x_2) \stackrel{\text{def}}{=} 1 - x_1 \cdot x_2$. If U_1, U_2 have noise-levels $\leq \beta$, then $\mathbf{U}^* := \mathsf{HTDF.Eval}^{in}(\mathsf{NAND}, (x_1, \mathbf{U}_1), (x_2, \mathbf{U}_2))$ will have a noise-level $\beta^* \leq (m+1)\beta$. Therefore if we compute $\mathbf{U}^* := \mathsf{HTDF.Eval}^{in}(g, (x_1, \mathbf{U}_1), \ldots, (x_\ell, \mathbf{U}_\ell))$ 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 $\mathbf{U}^* := \mathsf{HTDF.Eval}^{in}(g, (x_1, \mathbf{U}_1), \ldots, (x_\ell, \mathbf{U}_\ell))$ 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 Fully Homomorphic Signatures (Single Dataset)

Roadmap. We now show how to construct fully homomorphic signatures from HTDFs as a black box. We do so in several stages.

We begin by defining and constructing homomorphic signatures that can only be used to sign a single dataset. We also initially only consider selective security, where the data to be signed is chosen by an attacker prior to seeing the public parameters of the scheme. In Section 4.2 we show how to construct such schemes in the standard model, albeit with large public parameters whose size exceeds the maximal size of the dataset to be signed. The public parameters are just uniformly random and therefore, in the random oracle model, we can easily compress them to get a scheme with short public parameters. We also show that the latter scheme can be proven secure in the standard model under a simple-to-state and falsifiable (but non-standard) assumptions on hash functions.

In Section 4.4 we then show a generic transformation that combines a homomorphic signature scheme with selective security and an HTDF to get a homomorphic signature scheme with full security. Finally, in Section 5 we define multi-data signatures where the signer can sign many different datasets under different labels. We give a generic transformation from single-data homomorphic signatures to multi-data ones. Both of these transformations work in the standard model and preserve the efficiency of the underlying scheme. (In Appendix A, we also give an alternate transformation which yields a simpler construction of a multi-data scheme with full security in the RO model.) Lastly, in Section 6 we show how to make the signature schemes context hiding.

4.1 Definition

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

- $\mathsf{prms} \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N)$: Gets the security parameter λ and a data-size bound N. Generates public parameters prms. The security parameter also defines the message space $\mathcal{X}$.
- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$: Gets the security parameter λ. Generates a verification/secret keys pk, sk.
- $(\sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk}(x_1, \ldots, x_N)$: Signs some data $(x_1, \ldots, x_N) \in \mathcal{X}^N$.
- $\sigma^* \leftarrow \mathsf{SignEval}_{\mathsf{prms}}(g, ((x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell)))$: Homomorphically computes a signature σ^*.
- $\alpha_g \leftarrow \mathsf{Process}_{\mathsf{prms}}(g)$: Homomorphically computes a "public-key" α_g for the function g from the public parameters.
- $\mathsf{Verify}_{pk}(\alpha_g, y, \sigma)$: Verifies that y is indeed the output of g by checking the signature σ against α_g. We use these algorithms to implicitly define the "combined verification procedure": $\mathsf{Verify}^*_{pk}(g, y, \sigma) : \{$ Compute $\alpha_g \leftarrow \mathsf{Process}_{\mathsf{prms}}(g)$ and output $\mathsf{Verify}_{pk}(\alpha_g, y, \sigma)\}$.

We can think of $\mathsf{Process}, \mathsf{Verify}$ as a component of the combined verification procedure Verify^*, but it will be useful to define them separately. In particular, we will think of the $\mathsf{Process}$ algorithm as "pre-processing" a function g. The computational complexity of this step can depend on the circuit size of g but it can be performed *offline* prior to seeing the signature σ or even the verification key pk. The public-key α_g for the function g can be small and the "online verification" procedure $\mathsf{Verify}_{pk}(\alpha_g, y, \sigma)$ can be fast, with size/time independent of g.

Signing Correctness. Let $\mathsf{id}_i : \mathcal{X}^N \to \mathcal{X}$ be a canonical description of the function $\mathsf{id}_i(x_1, \ldots, x_N) \stackrel{\text{def}}{=} x_i$ (i.e., a circuit consisting of a single wire taking the i'th input to the output.) We require that any $\mathsf{prms} \in \mathsf{PrmsGen}(1^\lambda, 1^N)$, any $(pk, sk) \in \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$, 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 $\mathsf{prms} \in \mathsf{PrmsGen}(1^\lambda, 1^N)$, any $(pk, sk) \in \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$, 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:

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

where $\sigma^* \leftarrow \mathsf{SignEval}_{\mathsf{prms}}(g, ((x_1, \sigma_1), \ldots, (x_N, \sigma_N)))$. Moreover, we require correctness for *composed 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 $(\sigma_1, \ldots, \sigma_\ell)$:

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

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 $h_i \stackrel{\text{def}}{=} \mathsf{id}_i$.

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 $\sigma^* := \mathsf{SignEval}_{\mathsf{prms}}(g, ((x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell)))$ 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 $\beta_i \leq \beta_{init}$, then σ^* will have noise-level $\beta^* \leq \beta_{max}$.

Security Game. We define the security of homomorphic signatures via the following game between an attacker $\mathcal{A}$ and a challenger:

- The challenger samples $\mathsf{prms} \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N)$ and $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$ and gives prms, pk to the adversary.
- The attacker $\mathcal{A}(1^\lambda)$ chooses data $(x_1, \ldots, x_N) \in \mathcal{X}^*$ and sends it to the challenger.
- The challenger computes $(\sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk}(x_1, \ldots, x_N)$ and gives the signatures $(\sigma_1, \ldots, \sigma_N)$ 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 say a homomorphic signature scheme is *fully secure* if for all PPT $\mathcal{A}$, we have $\Pr[\mathcal{A} \text{ wins }] \leq \mathsf{negl}(\lambda)$ in the above game.

Remarks. We point out some extensions and relaxations of the definition that we also consider in this work.

- **Selective Security.** We will also consider a *selective security* game for single-data homomorphic signatures, where the attacker chooses the data $x_1, \ldots, x_N$ to be signed before seeing prms and pk. This is a natural security notion for the typical use-case where the user samples prms, pk, sk and signs the data in one step and therefore the data will not depend on prms, pk. We first show our basic construction satisfying *selective security*. We then show a generic transformation from a *selectively secure* scheme to a scheme satisfying *full security*.
- **Adaptive Individual Data Item Queries.** It is possible to extend the syntax of homomorphic signature schemes to also allow the user to sign different data items x_i individually with respect to their position i, rather than having to specify the entire dataset vector $(x_1, \ldots, x_N)$ all at at once. It is easy to see that our construction in Section 4.2 allows this. In this case, we can also extend our definition of full adaptive security to allow an adversary to query for

signatures of individual data items x_i adaptively, after seeing the signatures of other items. It is easy to see that our transformation from selective to fully adaptive security in Section 4.4 achieves this notion of security (with minimal syntactic changes to the proof). A similar extension can also be added to the setting of multiple-data sets as defined in Section 5. In this case, the user would be able to sign an individual data-item x_i with respect to position i of a dataset with label τ. Again it is easy to see that our construction in Section 5 allows for this and achieves full adaptive security in this setting.

- **Verification and Admissible Functions.** We note that, under the above definition, security 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 Basic Construction

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

- $\mathsf{prms} \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N)$: Choose $v_1, \ldots, v_N$ by sampling $v_i \xleftarrow{\$} \mathcal{V}$. Output $\mathsf{prms} = (v_1, \ldots, v_N)$.
- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$: Choose $(pk', sk') \leftarrow \mathsf{HTDF.KeyGen}(1^\lambda)$ and set $pk = pk'$, $sk = (\mathsf{prms}, sk')$.
- $(\sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk}(x_1, \ldots, x_N)$: Sample $u_i \leftarrow \mathsf{Inv}_{sk', x_i}(v_i)$ and set $\sigma_i := u_i$ for $i \in [N]$.
- $\sigma^* = \mathsf{SignEval}_{pk}(g, (x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell))$: Run $\mathsf{HTDF.Eval}^{in}_{pk'}$ procedure of the HTDF.
- $\alpha_g \leftarrow \mathsf{Process}_{\mathsf{prms}}(g)$: Compute $\alpha_g := \mathsf{HTDF.Eval}^{out}_{pk'}(g, v_1, \ldots, v_N)$.
- $\mathsf{Verify}_{pk}(\alpha_g, y, \sigma)$: If $f_{pk', y}(\sigma) = \alpha_g$ 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 pre-processing step $\mathsf{Process}_{\mathsf{prms}}(g)$ only needs to read the corresponding values v_i from the public parameters. The run-time of this step can therefore be sub-linear in N and only depends on the size of the circuit g (ignoring unused input wires). (IV) The efficiency of the scheme is inherited from that of the HTDF. Note that, although the pre-processing step $\mathsf{Process}_{\mathsf{prms}}(g)$ requires running $\mathsf{HTDF.Eval}$, the online verification step can be much more efficient. For our HTDF construction, it will only depend on the size of σ, α_g which only scale with the depth but not the size of the circuit g.

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 single-data security of homomorphic signatures.*

Proof. Assume that an adversary $\mathcal{A}$ has a non-negligible advantage in the single-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 $(pk', sk') \leftarrow \mathsf{HTDF.KeyGen}(1^\lambda)$, $v_i \leftarrow \mathcal{V}$ 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.[8] 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 $y := g(x_1, \ldots, x_N)$, $\alpha_g := \mathsf{HTDF.Eval}^{out}_{pk'}(g, v_1, \ldots, v_N)$, $u := \mathsf{HTDF.Eval}^{in}_{pk'}(g, (x_1, \sigma_1), \ldots, (x_N, \sigma_N))$. Then, since the signature σ' verifies, we have $f_{pk', y'}(u') = \alpha_g$. On the other hand, since g is an admissible function, the correctness of homomorphic evaluation ensures that $f_{pk', y}(u) = \alpha_g$. 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. $\square$

4.3 A Scheme with Short Public Parameters

We can adapt the above construction to get a scheme with short public parameters in the random oracle model. Instead of choosing $\mathsf{prms} = (v_1, \ldots, v_N)$, with $v_i \overset{\$}{\leftarrow} \mathcal{V}$ taken uniformly at random from the output space of the HTDF, we can set $\mathsf{prms} = r$ for some small $r \overset{\$}{\leftarrow} \{0,1\}^\lambda$ and implicitly define $v_i = H(r, i)$ where $H : \{0,1\}^* \to \mathcal{V}$ is a hash function modeled as a random oracle. The rest of the algorithms remain unchanged. It is easy to see that the same security proof as above goes through for this scheme in the random-oracle mode.

Moreover, we can define a standard-model assumption on the hash function H under which we can prove the above scheme secure. The assumption is falsifiable and simple-to-state, but it is not a standard assumption. It ties together the security of the hash function H with that of the underlying HTDF $\mathcal{F}$.

[8]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.

Definition 4.2. *Let $\mathcal{F} = (\mathsf{HTDF.KeyGen}, f, \mathsf{Inv}, \mathsf{HTDF.Eval}^{in}, \mathsf{HTDF.Eval}^{out})$ be an HTDF with index-space $\mathcal{X}$, input space $\mathcal{U}$, output space $\mathcal{V}$ and an input distribution $D_{\mathcal{U}}$. Let $H : \{0,1\}^* \to \mathcal{V}$ be a hash function. We say that H is* inversion unhelpful *for $\mathcal{F}$ if for any PPT adversary $\mathcal{A}$ the probability of $\mathcal{A}(1^\lambda)$ winning the following game is negligible in λ:*

- *Adversary $\mathcal{A}$ chooses values $x_1, \dots, x_N$ with $x_i \in \mathcal{X}$.*
- *Challenger chooses $(pk, sk) \leftarrow \mathsf{HTDF.KeyGen}(1^\lambda)$, $r \leftarrow \{0,1\}^\lambda$. For $i = 1, \dots, N$, it computes $v_i = H(r, i)$, $u_i \leftarrow \mathsf{Inv}_{sk,x_i}(v_i)$. It gives $(pk, r, u_1, \dots, u_N)$ to $\mathcal{A}$.*
- *Adversary $\mathcal{A}$ outputs $u, u' \in \mathcal{U}$ and $x \neq x' \in \mathcal{X}$ and wins if $f_{pk,x}(u) = f_{pk,x'}(u')$.*

Essentially, the above says that H is *inversion unhelpful* for $\mathcal{F}$ is seeing the inverses $u_i \leftarrow \mathsf{Inv}_{sk,x_i}(H(r,i))$ for x_i chosen adversarially but non-adaptively and r random, does not help the attacker in coming up with a claw for $\mathcal{F}$.

It is easy to see that the random-oracle construction outlined above is secure as long as the function H is *inversion unhelpful* for $\mathcal{F}$. The proof follows the proof of security of our basic construction in Section 4.2, with the only difference that the reduction sets the values $v_i = H(r,i)$ and the signatures $\sigma_i = u_i$ by getting r and $\{u_i\}$ them from the challenger in the above inversion-unhelpful security game.

It is also easy to see that if H is modeled as random oracle, then it is inversion-unhelpful for any HTDF $\mathcal{F}$. This is because we can "program" the outputs $H(r,i)$ to values $v_i = f_{pk,x_i}(u_i)$ for $u_i \leftarrow D_{\mathcal{U}}$ and then invert v_i to u_i without knowing the secret key sk of the HTDF. Therefore, seeing such inverses u_i cannot help the adversary in finding a claw.

We note that the above inversion-unhelpful assumption is simpler to state than simply assuming the resulting signature scheme is secure. In particular, the assumption does not depend on any homomorphic properties, the adversary does not get to choose a function of his choice, the challenger does not have to perform any homomorphic evaluations etc. Also, having such an assumption sets some contrast between the above homomorphic signature scheme with short parameters in the Random-Oracle model and a generic construction of homomorphic signatures based on SNARKs in the Random-Oracle model (see introduction). For the latter, we either need to rely on SNARK security, which is not a falsifiable assumption, or we could instead simply assume that the full signature scheme construction in the random-oracle model is secure but, since the construction of SNARKs is complex and involves heavy PCP machinery, this assumptions would be far from simple-to-state.

4.4 From Selective Security to Full Security

We now show how to construct a fully secure homomorphic signature scheme from any selectively secure scheme and an HTDF. On a high level, the transformation is similar to the use of *chameleon hashing* to go from security against selective chosen-message queries (e.g., the signing queries are all made ahead of time) to adaptive chosen-message security [KR00]. However, to make this work with homomorphic signatures, we would need the chameleon hash to also be homomorphic. Fortunately, HTDFs can be thought of as providing exactly such primitive.

In more detail, to sign some data $x_1, \dots, x_N$ under the new signature scheme, we first choose random values $v_1, \dots, v_N$ from the output of the HTDF, then we sign the values v_i under the selectively secure scheme to get signatures $\overline{\sigma}_i$ and finally we compute $u_i \leftarrow \mathsf{Inv}_{sk,x_i}(v_i)$ and give out the signature $\sigma_i = (v_i, u_i, \overline{\sigma}_i)$. To homomorphically evaluate some function g over such signatures we (1) run the input/output homomorphic computation the HTDF to compute values u^*, v^* so that u^* is an "opening" of v^* to the message $g(x_1, \ldots, x_N)$ and (2) we run the homomorphic evaluation of the signature scheme to compute a signature $\overline{\sigma}^*$ which certifies that v^* was computed correctly.

Security comes from the fact that values v_i signed under the selectively secure signature scheme are chosen uniformly at random and therefore non-adaptively. By the selective security of the underlying signature, this shows that an attacker cannot give the "wrong" v^* in his forgery. On the other hand, by the claw-free security of the HTDF, the attacker also cannot open the "right" v^* to the "wrong" message as this would produce a claw on the HTDF.

Construction. Let $\mathcal{F} = (\mathsf{HTDF.KeyGen},\ f,\ \mathsf{Inv},\ \mathsf{HTDF.Eval}^{in},\ \mathsf{HTDF.Eval}^{out})$ be an HTDF with index-space $\mathcal{X}$, input space $\mathcal{U}$, output space $\mathcal{V}$ and an input distribution $D_{\mathcal{U}}$. Let $\mathcal{S}' = (\mathsf{PrmsGen}', \mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Verify}', \mathsf{Process}', \mathsf{SignEval}')$ be a selectively secure homomorphic signature scheme. Without loss of generality and for simplicity of exposition, we will assume that the message space of $\mathcal{S}'$ is the same as the output space $\mathcal{V}$ of the HTDF (we can always represent the values $v \in \mathcal{V}$ as bits and sign them bit-by-bit if this is not the case). For a function $g : \mathcal{X}^\ell \to \mathcal{X}$ we define a corresponding function $g' : \mathcal{V}^\ell \to \mathcal{V}$ as $g'(v_1, \ldots, v_\ell) = \mathsf{HTDF.Eval}^{out}(g, v_1, \ldots, v_\ell)$.

- $\mathsf{prms} \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N)$: Use the selectively secure signature scheme $\mathsf{prms} \leftarrow \mathsf{PrmsGen}'(1^\lambda, 1^N)$.
- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$: Choose

  $$(pk_h, sk_h) \leftarrow \mathsf{HTDF.KeyGen}(1^\lambda) \quad , \quad (pk', sk') \leftarrow \mathsf{KeyGen}'(1^\lambda, \mathsf{prms}).$$

  Set $pk = (pk_h, pk'), sk = (sk_h, sk')$.
- $(\sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk'}(x_1, \ldots, x_N)$:
  - For each $i \in [N]$: sample $v_i \leftarrow \mathcal{V}$, $u_i \leftarrow \mathsf{Inv}_{sk_h, x_i}(v_i)$.
  - Compute $(\bar{\sigma}_1, \ldots, \bar{\sigma}_N) \leftarrow \mathsf{Sign}'_{sk'}(v_1, \ldots, v_N)$.
  - Set $\sigma_i = (v_i, u_i, \overline{\sigma}_i)$.
- $\sigma^* = \mathsf{SignEval}_{pk}(g, (x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell))$: Parse $\sigma_i = (v_i, u_i, \overline{\sigma}_i)$.
  - Compute $v^* := \mathsf{HTDF.Eval}^{out}_{pk_h}(g, v_1, \ldots, v_\ell)$, $u^* := \mathsf{HTDF.Eval}^{in}_{pk_h}(g, (x_1, u_1), \ldots, (x_\ell, u_\ell))$.
  - Compute $\overline{\sigma}^* = \mathsf{SignEval}'_{pk'}(g', (v_1, \overline{\sigma}_1), \ldots, (v_\ell, \overline{\sigma}_\ell))$.
  - Output $\sigma^* = (v^*, u^*, \overline{\sigma}^*)$,
- $\alpha_g \leftarrow \mathsf{Process}_{\mathsf{prms}}(g)$: Compute $\alpha_g := \mathsf{Process}'_{\mathsf{prms}}(g')$.
- $\mathsf{Verify}_{pk}(\alpha_g, y, \sigma^*)$: Parse $\sigma^* = (v^*, u^*, \overline{\sigma}^*)$. Verify that $f_{pk_h, y}(u^*) = v^*$ and $\mathsf{Verify}'_{pk'}(\alpha_g, v^*, \overline{\sigma}^*) =$ `accept`: if both conditions hold then accept, else reject.

Correctness. The correctness of the signature scheme follows readily from the correctness of $\mathcal{S}'$ and $\mathcal{F}$. A function g is admissible on values $x_1, \ldots, x_N$ under the scheme $\mathcal{S}$ if it is admissible under the HTDF $\mathcal{F}$ and if the corresponding function g' is also admissible over all values $(v_1, \ldots, v_n) \in \mathcal{V}^N$ under the selectively-secure signature scheme $\mathcal{S}'$.

Theorem 4.3. *If $\mathcal{F}$ is a secure (leveled) HTDF and $\mathcal{S}'$ is a selectively secure (leveled) homomorphic signature scheme, then the above construction of $\mathcal{S}$ is a fully secure (leveled) homomorphic signature scheme.*

Proof. Let $\mathcal{A}$ be some adversary in the adaptive signature security game against the scheme $\mathcal{S}$. Let (g, y', σ') denote the adversary's forgery at the end of the game, and parse $\sigma' = (v', u', \overline{\sigma}')$. Let $x_1, \ldots, x_N$ denote the messages chosen by the adversary in the game and $v_1, \ldots, v_N$ be the values contained in the signatures that it gets back. Let E_1 be the event that $\mathcal{A}$ wins the game *and* $v' = \mathsf{HTDF.Eval}^{out}_{pk_h}(g, v_1, \ldots, v_N)$. Let E_2 be the even that $\mathcal{A}$ wins the game $v' \neq \mathsf{HTDF.Eval}^{out}_{pk_h}(g, v_1, \ldots, v_N)$. The probability that $\mathcal{A}$ wins the game is $\Pr[E_1 \vee E_2] \leq \Pr[E_1] + \Pr[E_1]$.

Firstly, we show that $\Pr[E_1]$ is negligible. We do this via a reduction breaking HTDF security of $\mathcal{F}$ with probability $\Pr[E_1]$. The reduction gets an HTDF public key pk_h. It chooses $(\mathsf{prms}, pk', sk')$ for the selectively-secure signature scheme on its own and simulates the signature game for $\mathcal{A}$ with one modification: to answer the signing query, instead of choose $v_i \leftarrow \mathcal{V}$, $u_i \leftarrow \mathsf{Inv}_{sk_h, x_i}(v_i)$ it chooses $u_i \leftarrow D_{\mathcal{U}}$ and $v_i = f_{pk_h, x_i}(u_i)$. This change is statistically indistinguishable by the "Distributional Equivalence of Inversion" property of the HTDF. Finally, assume that $\mathcal{A}$ outputs a forgery causing E_1 to occur. Let $y = g(x_1, \ldots, x_N)$ and let $u = \mathsf{HTDF.Eval}^{in}_{pk_h}((x_1, u_1), \ldots, (x_N, u_N))$. Then $y \neq y'$ and $f_{pk,y}(u) = f_{pk,y'}(u')$. Therefore the tuple (u, u', y, y') allows the reduction to break HTDF security.

Secondly, we show that $\Pr[E_2]$ is negligible. We do this via reduction breaking the selective security of $\mathcal{S}'$ with $\Pr[E_2]$. The reduction starts by (non-adaptively) choosing random messages $v_1, \ldots, v_N$ with $v_i \leftarrow \mathcal{V}$ and giving them to its challenger. It gets back prms, pk' and $\overline{\sigma}_1, \ldots, \overline{\sigma}_N$. The reduction chooses its own keys (pk_h, sk_h) for the HTDF and gives $(\mathsf{prms}, (pk_h, pk'))$ to $\mathcal{A}$. The adversary $\mathcal{A}$ replies with messages $(x_1, \ldots, x_N)$ and the reduction computes $u_i \leftarrow \mathsf{Inv}_{sk_h, x_i}(v_i)$ and gives back the values $\sigma_i = (v_i, u_i, \overline{\sigma}_i)$ to $\mathcal{A}$. Finally, assume that $\mathcal{A}$ outputs a forgery causing E_1 to occur. Then $(g', v', \overline{\sigma}')$ is a forgery against $\mathcal{S}'$ since $v' \neq g'(v_1, \ldots, v_N)$.

Combining the above, this shows that the probability that $\mathcal{A}$ wins the adaptive signature security game against $\mathcal{S}$ is negligible, and the theorem follows.

□

5 Multi-Data Homomorphic Signatures

We now define and construct *multi-data* homomorphic signatures. In such a scheme, the signer can sign many different datasets of arbitrary size. Each dataset is tied to some labels τ_i (e.g., the name of the dataset) and the verifier is assumed to know the label of the dataset over which he wishes to verify computation. In Section 5.2, we show how to construct multi-data homomorphic signatures starting from a single dataset scheme. In Appendix A, we show another transformation which enjoys efficiency improvements and supports signing of unbounded datasets starting from single dataset scheme with short public parameters (such as our construction in the random oracle model 4.3).

5.1 Definition

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

- $\mathsf{prms} \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N)$: Gets the security parameter λ and a data-size bound N. Generates public parameters prms.
- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$: produces a public verification key pk and a secret signing key sk.
- $(\sigma_\tau, \sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk}((x_1, \ldots, x_N), \tau)$: Signs some data $\bar{x} \in \mathcal{X}^*$ under a label $\tau \in \{0,1\}^*$.
- $\sigma^* = \mathsf{SignEval}_{\mathsf{prms}}(g, \sigma_\tau, (x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell))$: Homomorphically computes the signature σ^*.
- $\alpha_g \leftarrow \mathsf{Process}_{\mathsf{prms}}(g)$: Produces a "public-key" α_g for the function g.
- $\mathsf{Verify}_{pk}(\alpha_g, y, \tau, (\sigma_\tau, \sigma^*))$: Verifies that $y \in \mathcal{X}$ is indeed the output of the function g over the data signed with label τ. We define the "combined verification procedure":
  $\mathsf{Verify}^*_{pk}(g, y, \tau, \sigma_\tau, \sigma^*) : \{$ Compute $\alpha_g \leftarrow \mathsf{Process}_{\mathsf{prms}}(g)$ and output $\mathsf{Verify}_{pk}(\alpha_g, y, \tau, (\sigma_\tau, \sigma^*))\}$.

Correctness. The correctness requirements are analogous to those of the single-data definition. We right away define correctness of evaluation to allow for *composed evaluation.*

Correctness of Signing. We require that any $\mathsf{prms} \in \mathsf{PrmsGen}(1^\lambda, 1^N)$, $(pk, sk) \in \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$, 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 $\mathsf{Verify}^*_{pk}(\mathsf{id}_i, x_i, \tau, (\sigma_\tau, \sigma_i)) = \texttt{accept}$. 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 $\sigma_\tau, (\sigma_1, \ldots, \sigma_\ell)$:

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

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.

Multi-Data Security. We define the security via the following game between an attacker $\mathcal{A}$ and a challenger:

- The challenger samples $\mathsf{prms} \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N)$, $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$ and gives prms, 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 $\mathsf{Verify}^*_{pk}(g, \tau, y', (\sigma'_\tau, \sigma')) = \texttt{accept}$ 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 From Single-Data to Multi-Data

We now describe our transformation from a single-data homomorphic signature scheme to a multi-data scheme. We sample the public parameters of the single-data homomorphic signature scheme once and for all, then for each signing query we sample a pair of public/secret keys of the homomorphic scheme (using the same public parameters). The dataset information along with the public key are signed using the regular homomorphic signature scheme. The dataset itself is signed using the sampled secret key. To verify the authenticity, it is sufficient to verify the dataset information with the public key, and authenticate the output of the function with respect to this public key.

Let $\mathcal{S}' = (\mathsf{PrmsGen}', \mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Verify}', \mathsf{Process}', \mathsf{SignEval}')$ be a fully secure one-dataset homomorphic signature scheme. Let $\mathcal{S}^{nh} = (\mathsf{NH.KeyGen}, \mathsf{NH.Sign}, \mathsf{NH.Verify})$ be any standard (not homomorphic) signature scheme. We construct a multi-data homomorphic signature scheme $\mathcal{S} = (\mathsf{PrmsGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Process}, \mathsf{SignEval})$ with message space $\mathcal{X}$ as follows.

- $\mathsf{prms} \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N)$: Sample and output parameters of the single-data homomorphic signature scheme: $\mathsf{prms} \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N)$.
- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$: Choose $(pk_1, sk_1) \leftarrow \mathsf{NH.KeyGen}(1^\lambda, \mathsf{prms})$. Samp set $pk = pk_1$, $sk = (sk_1, \mathsf{prms})$.
- $(\sigma_\tau, \sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk}((x_1, \ldots, x_N), \tau)$:
  - Sample secret and public keys of the single-data homomorphic signature scheme: $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$.
  - Sign the dataset size, the tag and the public key of the single-data homomorphic signature scheme using non-homomorphic scheme: $\rho \leftarrow \mathsf{NH.Sign}_{sk_1}((pk_2, \tau, N))$. Set $\sigma_\tau = (pk_2, \tau, N, \rho)$.
  - Sign the dataset using the single-data homomorphic scheme: $(\sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}'_{sk_2}(x_1, \ldots, x_N)$.
  - Output $(\sigma_\tau, \sigma_1, \ldots, \sigma_N)$.
- $\sigma^* = \mathsf{SignEval}_{pk}(g, \sigma_\tau, \sigma(x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell))$: Parse $\sigma_\tau = (pk_2, \tau, N, \rho)$. Apply the single-data evaluation algorithm. $\sigma^* = \mathsf{SignEval}_{\mathsf{prms}}(g, (x_1, \sigma_1), \ldots, (x_N, \sigma_N))$.
- $\alpha_g \leftarrow \mathsf{Process}_{\mathsf{prms}}(g)$: Output $\alpha_g \leftarrow \mathsf{Process}'_{\mathsf{prms}}(g)$.
- $\mathsf{Verify}_{pk}(\alpha_g, y, \tau, (\sigma_\tau, \sigma^*))$: Parse $\sigma_\tau = (\mathsf{prms}, pk_2, \tau, N, \rho)$ and accept if and only if the following two conditions are satisfied:
  1. Verify the parameters of the single-data homomorphic scheme's public key and the dataset:
     $\mathsf{NH.Verify}_{pk_1}((pk_2, \tau, N), \rho) = \texttt{accept}$, and
  2. Verify the homomorphically computed signature: $\mathsf{Verify}'_{pk_2}(\alpha_g, y, \sigma^*) = \texttt{accept}$.

Correctness. Correctness of the scheme follows from the correctness of the regular signature scheme and single-data homomorphic scheme.

Security. The security follows from two main observations: first, no adversary is able to fake the parameters or the public key of the single-data homomorphic signature due to security of the standard signature scheme. Given that, no adversary is able to fake the result of the the computation due to the security of the single-data homomorphic signature scheme. In particular, we first switch to Game 1, where the adversary looses if it is able to make a forgery on some of the dataset information signed under the regular scheme: (pk_2, τ, N). Now, if there is an adversary able to win Game 1, then we can convert it to an adversary that breaks the security of the homomorphic scheme. The adversary takes (prms, pk_2) as a part of the challenge. Guesses an index j^* and sets $pk_{2,j^*} = pk_2$. It then asks the challenger to sign the data $(x_1, \ldots, x_N)$ at query j^* and signs all other datasets by itself by sampling a pair of public/secret keys. A forgery of type II can then be used to break the security of single-data homomorphic scheme.

Theorem 5.1. *Assume $\mathcal{S}'$ is a fully secure single-data homomorphic signature scheme and $\mathcal{S}^{nh}$ is a regular signature scheme. Then, $\mathcal{S}$ is a fully secure many-dataset homomorphic signature scheme.*

Proof. We prove the security via a series of indistinguishable games.

- Let **Game 0** be the multi-data security game.
- Let **Game 1** be the modified version of **Game 1**, except the attacker loses the game if it outputs a non-homomorphic forgery. That is, a forgery of the form $(g, y, \tau, (\sigma_\tau = (pk_2, \tau, N', \rho), \sigma^*)$, where:

  1. $\tau \neq \tau_j$ for any j, or
  2. $\tau = \tau_j$ for some j but $N' \neq N_j$ or
  3. $\tau = \tau_j$ for some j, $N' = N_j$, but $pk_2 \neq pk_{2,j}$, where $pk_{2,j}$ is the public key used for single-data signature scheme.

  That is, if the parameters of the dataset are invalid, the adversary losses the game. Note that this is the superset of type I forgeries. Clearly, if there exists a winning adversary in **Game 1**, then we can break the security of the regular signature scheme, since we obtain a valid signature ρ of (pk_2, τ, N) which was never signed before.

Now, assume there exists an adversary $\mathcal{A}$ that wins in **Game 1**. Then, it must be able to come up with a type II forgery. Hence, we can construct an adversary $\mathcal{A}'$ that breaks the security of the single-data homomorphic signature scheme. $\mathcal{A}'$ receives parameters prms, pk_2 as the challenge, it then generates parameters of the regular signature scheme $(pk_1, sk_1) \leftarrow \mathsf{NH.KeyGen}(1^\lambda)$ and forwards $\mathsf{prms}, pk = pk_1$ to $\mathcal{A}$. It also chooses an index j^* of the dataset on which it guesses the type II will be made and sets $pk_{2,j^*} = pk_2$. When $\mathcal{A}$ asks to sign a dataset $j = j^*$ with items $(x_1, \ldots, x_N)$, $\mathcal{A}'$ forwards it to the challenger to obtain the signatures $(\sigma_1, \ldots, \sigma_N)$. It signs (pk_{2,j^*}, τ, N) to obtain σ_τ using sk_1 and forwards $(\sigma_\tau, \sigma_1, \ldots, \sigma_N)$ to $\mathcal{A}$. All other datasets $j \neq j^*$ it signs honestly by generating the public/secret keys of the single-data homomorphic signature scheme. Finally, suppose $\mathcal{A}$ outputs $(g, \tau, y, (\sigma_\tau = (pk_2, \tau, N', \rho), \sigma^*)$ of type II forgery. Then, assuming $\mathcal{A}'$ guessed the dataset correctly, we know that $\tau = \tau_{j^*}, N = N_{j^*}$ and $pk_2 = pk_{2,j^*}$, $\mathsf{Verify}^*_{pk_2}(g, y, \sigma^*) =$

`accept` but $g \neq g(x_1, \ldots, x_N)$. Hence, $\mathcal{A}'$ can output (g, y, σ^*) to break the security of single-data homomorphic signature scheme. This shows that if the regular signature scheme is secure and the single-data homomorphic signature scheme is secure, then $\mathcal{S}$ is also secure. $\square$

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. The "hiding" signature $\widetilde{\sigma}$ can be simulated given only g, y no matter which original signature σ was used to create it. One additional advantage of this procedure is that it also compresses the size of the signature from $m^2 \log q$ bits needed to represent σ to $O(m \log q)$ bits needed to represent $\widetilde{\sigma}$. However, once the hiding procedure is applied, the signatures no longer support additional homomorphic operations on them.

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

- *Correctness: For any* $\mathsf{prms} \in \mathsf{PrmsGen}(1^\lambda, 1^N), (pk, sk) \in \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$ *and any* α, y, σ *such that* $\mathsf{Verify}_{pk}(\alpha, y, \sigma) = \mathtt{accept}$*, for any* $\widetilde{\sigma} \in \mathsf{Hide}_{pk,y}(\sigma)$ *we have* $\mathsf{HVerify}_{pk}(\alpha, y, \widetilde{\sigma}) = \mathtt{accept}$.

- *Unforgeability: Single-data signature security holds when we replace the* Verify *procedure by* $\mathsf{HVerify}$ *in the security game.*

- *Context-Hiding Security: There is a simulator* Sim *such that, for any* fixed *(worst-case) choice of* $\mathsf{prms} \in \mathsf{PrmsGen}(1^\lambda, 1^N), (pk, sk) \in \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$ *and any* α, y, σ *such that* $\mathsf{Verify}_{pk}(\alpha, y, \sigma) = \mathtt{accept}$ *we have:* $\mathsf{Hide}_{pk,y}(\sigma) \approx \mathsf{Sim}(sk, \alpha, y)$ *where the randomness is only over the random coins of the simulator and the* Hide *procedure.*[9] *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* $\widetilde{\sigma} \leftarrow \mathsf{Hide}_{pk,x}(\sigma)$*,* $\mathsf{HVerify}_{pk}(g, \mathsf{Process}(g), y, \tau, (\sigma_\tau, \sigma))$ *such that:*

[9]Since $pk, sk, \alpha, y, \sigma$ are fixed, indistinguishability holds even if these values are known to the distinguisher.

- *Correctness: For any* $\mathsf{prms} \in \mathsf{PrmsGen}(1^\lambda, 1^N), (pk, sk) \in \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$ *and any* $\alpha, y, \sigma_\tau, \sigma$ *such that* $\mathsf{Verify}_{pk}(\alpha, y, \tau, (\sigma_\tau, \sigma)) = \mathtt{accept}$*, for any* $\widetilde{\sigma} \in \mathsf{Hide}_{pk,y}(\sigma)$ *we have*

$$\mathsf{HVerify}_{pk}(\alpha, y, \tau, (\sigma_\tau, \widetilde{\sigma})) = \mathtt{accept}$$

- *Unforgeability: Multi-data signature security holds when we replace the* Verify *procedure by* HVerify *in the security game.*

- *Context-Hiding Security: Firstly, in the procedure* $(\sigma_\tau, \sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk}(x_1, \ldots, x_N, \tau)$*, we require that* σ_τ *can only depend on* (sk, N, τ) *but not on the data* $\{x_i\}$*. Secondly, we require that there is a simulator* Sim *such that, for any* fixed *(worst-case) choice of* $\mathsf{prms} \in \mathsf{PrmsGen}(1^\lambda, 1^N), (pk, sk) \in \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$ *and any* $\alpha, y, \sigma, \sigma_\tau$ *such that* $\mathsf{Verify}_{pk}(\alpha, y, \tau, (\sigma_\tau, \sigma)) = \mathtt{accept}$ *we have:*

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

  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.*

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* $\tilde{u} \leftarrow \mathsf{HTDF.Hide}_{pk,x}(u)$ *and* $\mathsf{HTDF.Verify}_{pk}(\tilde{u}, x, v)$ *satisfying:*

- Correctness: *For any* $(pk, sk) \in \mathsf{KeyGen}(1^\lambda)$ *any* $u \in \mathcal{U}$*, any* $x \in \mathcal{X}$ *and any* $\tilde{u} \in \mathsf{HTDF.Hide}_{pk,x}(u)$ *we have* $\mathsf{HTDF.Verify}_{pk}(\tilde{u}, x, f_{pk,x}(u)) = \mathtt{accept}$.

- Claw-freeness on hidden inputs: *We augment standard HTDF security with the following requirement. For all PPT* $\mathcal{A}$ *we require:*

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

  In other words, if an attacker know u *such that* $f_{pk,x}(u) = v$ *then he cannot also produce* $\tilde{u}'$ *such that* $\mathsf{HTDF.Verify}_{pk}(\tilde{u}', x', v) = \mathtt{accept}$ *when* $x' \neq x$*.* [10]

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

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

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

[10] This implies standard HTDF security since any attacker that finds $x \neq x', u, u'$ such that $f_{pk,x}(u) = f_{pk,x'}(u')$ can also apply $\tilde{u}' \leftarrow \mathsf{Hide}_{pk,x'}(u')$ to break claw-freeness on hidden inputs.

From Context-Hiding HTDFs to Signatures. We can easily modify the signature schemes constructed in Section 4 (single-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 $\mathsf{HVerify}_{pk}(\alpha, y, \widetilde{\sigma})$ of the signature scheme (resp. $\mathsf{HVerify}_{pk}(\alpha, y, \tau, (\sigma_\tau, \widetilde{\sigma}))$ for a multi-data scheme) are defined the same ways as the original Verify procedures of the signature, *except* that, instead of checking $f_{pk,y}(\widetilde{\sigma}) = \alpha$ we now check $\mathsf{HTDF.Verify}_{pk}(\widetilde{\sigma}, y, \alpha) = \mathtt{accept}$. 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 proof of Theorem 4.1. 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 $\mathsf{HTDF.Verify}_{pk}(\widetilde{\sigma}, y', v^*) = \mathtt{accept}$ for $y' \neq y$.

Context-Hiding security of the signature scheme follows from that of the HTDF. We define the signature simulator $\mathsf{Sim}(sk, g, y, [\tau, \sigma_\tau])$ to compute the value $v^* = \mathsf{SignEval}^{in}_{pk}(g, v_1, \ldots, v_N)$ as is done by the verification procedure of the signature schemes. It then output $\widetilde{\sigma} \leftarrow \mathsf{HTDF.Sim}(sk, y, v^*)$. 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 $\tilde{u} \leftarrow \mathsf{HTDF.Hide}_{pk,x}(u)$ to be a NIZK-PoK with the statement v and witness u for the relation $f_{pk,x}(u) = v$. The $\mathsf{HTDF.Verify}$ procedure would simply verify the proof $\tilde{u}$. Claw-freeness follows from the PoK property and context-hiding follows from ZK.[11] 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.

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 $\mathbf{B} = \mathbf{AU} + y\mathbf{G}$ 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 $\mathbf{H} = [\mathbf{A} \mid \mathbf{AU} + y\mathbf{G}]$, where $\mathbf{G}$ is the matrix from part (3) of Lemma 2.2. Then:*

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

- *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. Firstly, we make the following modifications to the underlying HTDF construction.

- 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 $\mathbf{r} \overset{\$}{\leftarrow} \{0, 1\}^m$ and setting $\mathbf{z} = \mathbf{A} \cdot \mathbf{r}$ (and discarding $\mathbf{r}$).[12] Let $p_{extra}(n, m, \log q) = \mathsf{poly}(\lambda)$ be the polynomial form Lemma 6.4 and define $\tilde{\beta}_{max} = \beta_{max} \cdot p_{extra}(n, m, \log q)$.

In addition, we add the following procedures for context-hiding security.

- $\tilde{\mathbf{u}} \leftarrow \mathsf{HTDF.Hide}_{pk,x}(\mathbf{U})$: Let $\mathbf{V} = f_{pk,x}(\mathbf{U}) = \mathbf{AU} + x\mathbf{G}$. Set

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

  Note that $(2x - 1) \in \{-1, 1\} \neq 0$. Output $\tilde{\mathbf{u}} \leftarrow \mathsf{SampleRight}(\mathbf{H}, \mathbf{U}, \mathbf{z})$. Note that $\mathbf{H} \cdot \tilde{\mathbf{u}} = \mathbf{z}$ and $||\tilde{\mathbf{u}}||_\infty \leq \tilde{\beta}_{max}$.

- $\mathsf{HTDF.Verify}_{pk}(\tilde{\mathbf{u}}, x, \mathbf{V})$: Compute $\mathbf{H} := [\mathbf{A} \mid \mathbf{V} + (x - 1)\mathbf{G}]$. Check $||\tilde{\mathbf{u}}||_\infty \leq \tilde{\beta}_{max}$ and $\mathbf{H} \cdot \tilde{\mathbf{u}} = \mathbf{z}$. If so accept, else reject.

- For context-hiding security, we define $\mathsf{HTDF.Sim}(sk = \mathsf{td}, x, \mathbf{V})$ which computes $\mathbf{H} := [\mathbf{A} \mid \mathbf{V} + (x - 1)\mathbf{G}]$ and outputs $\tilde{\mathbf{u}} \leftarrow \mathsf{SampleLeft}(\mathbf{H}, \mathsf{td}, \mathbf{z})$.

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

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

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 $\mathbf{r} \overset{\$}{\leftarrow} \{0, 1\}^m$

[12] 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.

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

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

where the last equality follows since $x \neq x' \quad \Rightarrow \quad x + x' = 1$. 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:

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

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. $\square$

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. It also remains an open problem to come up with a (leveled) fully homomorphic signature scheme with short public parameters under a standard assumption without random oracles.

Acknowledgments. We thank Chris Peikert and Dario Catalano for helpful comments and suggestions.

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 Cramer [Cra12], pages 1–20.

[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.

[CFW14] Dario Catalano, Dario Fiore, and Bogdan Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 371–389. Springer, 2014.

[CG13] Ran Canetti and Juan A. Garay, editors. *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*. Springer, 2013.

[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.

[Cra12] Ronald Cramer, editor. *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*. Springer, 2012.

[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.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Canetti and Garay [CG13], pages 75–92.

[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.

[KR00] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *NDSS 2000*, San Diego, California, USA, February 2–4, 2000. The Internet Society.

[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 Canetti and Garay [CG13], pages 21–39.

[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.

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In Cramer [Cra12], pages 422–439.

[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.

A Another Single to Multi-Data Transformation

We describe another generic transformation from single-data homomorphic signature scheme with short public parameters prms (independent on the data size; realized by our construction in Section 4.3) to multi-data scheme. We point out that for this transformation it is sufficient to start with a selectively secure single data scheme leading efficiency improvements. However, the resulting construction does not work well for verifiable outsourcing since the verification algorithm runs in time proportional to the run-time of the function.

Let $\mathcal{S}' = (\mathsf{PrmsGen}', \mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Verify}', \mathsf{Process}', \mathsf{SignEval}')$ be a selectively secure homomorphic signature scheme. Let $\mathcal{S}^{nh} = (\mathsf{NH.KeyGen}, \mathsf{NH.Sign}, \mathsf{NH.Verify})$ be any standard (<u>n</u>ot <u>h</u>omomorphic) signature scheme. We construct a multi-data homomorphic signature scheme $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{SignEval})$ with message space $\mathcal{X}$ as follows.[13]

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$: Choose $(pk_1, sk_1) \leftarrow \mathsf{NH.KeyGen}(1^\lambda)$ set $pk = pk_1$, $sk = sk_1$.
- $(\sigma_\tau, \sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk}((x_1, \ldots, x_N), \tau)$:
  - Sample parameters and keys of the single-data homomorphic signature scheme: $\mathsf{prms} \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N), (pk_2, sk_2) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$.
  - Sign the dataset size, the tag and the public parameters of the single-data homomorphic signature scheme using non-homomorphic scheme: $\rho \leftarrow \mathsf{NH.Sign}_{sk_1}((\mathsf{prms}, pk_2, \tau, N))$. Set $\sigma_\tau = (\mathsf{prms}, pk_2, \tau, N, \rho)$.
  - Sign the dataset using the single-data homomorphic scheme: $(\sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}'_{sk_2}(x_1, \ldots, x_N)$.

[13]Note that we do not define a separate $\mathsf{PrmsGen}$ or $\mathsf{Process}$ algorithms, since this construction does not support efficient verification with preprocessing.

– Output $(\sigma_\tau, \sigma_1, \ldots, \sigma_N)$.

- $\sigma^* = \mathsf{SignEval}_{pk}(g, \sigma_\tau, \sigma(x_1, \sigma_1), \ldots, (x_\ell, \sigma_\ell))$: Parse $\sigma_\tau = (\mathsf{prms}, pk_2, \tau, N, \rho)$. Apply the single-data evaluation algorithm. $\sigma^* = \mathsf{SignEval}_{\mathsf{prms}}(g, (x_1, \sigma_1), \ldots, (x_N, \sigma_N))$.
- $\mathsf{Verify}_{pk}(g, y, \tau, (\sigma_\tau, \sigma^*))$: Parse $\sigma_\tau = (\mathsf{prms}, pk_2, \tau, N, \rho)$ and accept if and only if the following two conditions are satisfied:
  1. Verify the parameters of the single-data homomorphic scheme and the dataset: $\mathsf{NH.Verify}_{pk_1}((\mathsf{prms}, pk_2, \tau, N), \rho) = \texttt{accept}$, and
  2. Verify the homomorphically computed signature: $\mathsf{Verify}'_{pk_2}(g, \mathsf{Process}_{\mathsf{prms}}(g), y, \sigma^*) = \texttt{accept}$.

Remarks. We note that that there is no a-prior bound on the size of the datasets in this construction. However, since σ_τ includes the description of the public parameters of the single-data homomorphic scheme, these parameters must be small (as in our construction in the Random Oracle model).

Correctness. Correctness of the scheme follows readily from the correctness of the regular signature scheme and the single-data homomorphic signature scheme.

Security. On the high level, security follows from the fact that by the security property of the standard signature scheme, the adversary cannot modify the data size or the public parameters of the single-data signature scheme. And, given the correct parameters of the single-data signature scheme, we can efficiently verify the result of the computation by verifying the homomorphically computed signature.

Theorem A.1. *Assume $\mathcal{S}'$ is a selectively secure single-data homomorphic signature scheme and $\mathcal{S}^{nh}$ is a regular signature scheme. Then, $\mathcal{S}$ is a fully secure many-dataset homomorphic signature scheme.*

Proof. We prove the security via a series of indistinguishable games.

- Let **Game 0** be the multi-data security game.
- Let **Game 1** be the modified version of **Game 1**, except the attacker loses the game if it outputs a non-homomorphic forgery. That is, a forgery of the form $(g, y, \tau, (\sigma_\tau = (\mathsf{prms}, pk_2, \tau, N', \rho), \sigma^*)$, where:
  1. $\tau \neq \tau_j$ for any j, or
  2. $\tau = \tau_j$ for some j but $N' \neq N_j$ or
  3. $\tau = \tau_j$ for some j, $N' = N_j$, but $\mathsf{prms} \neq \mathsf{prms}_j$ or $pk_2 \neq pk_{2,j}$, where $\mathsf{prms}_j, pk_{2,j}$ are the parameters used for single-data signature scheme.

  That is, if the parameters of the dataset are invalid, the adversary losses the game. Note that this is the superset of type I forgeries. Clearly, if there exists a winning adversary in **Game 1**, then we can break the security of the regular signature scheme, since we obtain a valid signature ρ of $(\mathsf{prms}, pk_2, \tau, N')$ which was never signed before.

Now, assume there exists an adversary $\mathcal{A}$ that wins in **Game 1**. Then, it must be able to come up with a type II forgery. Hence, we can construct an adversary $\mathcal{A}'$ that breaks the security of the single-data homomorphic signature scheme. $\mathcal{A}'$ generates parameters of the regular signature scheme $(pk_1, sk_1) \leftarrow \mathsf{NH.KeyGen}(1^\lambda)$ and forwards $pk = pk_1$ to $\mathcal{A}$. It also chooses an index j^* of the dataset on which it guesses the type II will be made. When $\mathcal{A}$ asks to sign a dataset $j = j^*$ with items $(x_1, \ldots, x_N)$, $\mathcal{A}'$ forwards it to the single-data challenger to obtain the signatures $(\sigma_1, \ldots, \sigma_N)$ along with the public parameters $(\mathsf{prms}_{j^*}, pk_{2,j^*})$. It signs $(\mathsf{prms}_{j^*}, pk_{2,j^*}, \tau, N)$ to obtain σ_τ using sk_1 and forwards $(\sigma_\tau, \sigma_1, \ldots, \sigma_N)$ to $\mathcal{A}$. All other datasets $j \neq j^*$ it signs honestly by generating the parameters of the single-data homomorphic signature scheme. Finally, suppose $\mathcal{A}$ outputs $(g, \tau, y, (\sigma_\tau = (\mathsf{prms}, pk_2, \tau, N', \rho), \sigma^*)$ of type II forgery. Then, assuming $\mathcal{A}'$ guessed the dataset correctly, we know that $\tau = \tau_{j^*}, N = N_{j^*}, \mathsf{prms} = \mathsf{prms}_{j^*}$ and $pk_2 = pk_{2,j^*}$, $\mathsf{Verify}'_{pk_2}(g, \mathsf{Process}_{\mathsf{prms}}(g), y, \sigma^*) = \mathtt{accept}$ but $g \neq g(x_1, \ldots, x_N)$. Hence, $\mathcal{A}'$ can output (g, y, σ^*) to break the security of single-data homomorphic signature scheme. This shows that if the regular signature scheme is secure and the single-data homomorphic signature scheme is secure, then $\mathcal{S}$ is also secure. $\square$

B HTDFs and Fully Homomorphic Encryption

We briefly and informally sketch an interesting conceptual connection between fully homomorphic encryption and signatures. We show that both of these primitives can be constructed from a type of HTDFs: signatures correspond to *equivocable* HTDFs, whereas encryption corresponds to *extractable* HTDFs.

For equivocable HTDFs, which was the notion we defined in this paper, we wanted the output $v = f_{pk,x}(u)$ over a random u to statistically hide x and to have an "equivocation trapdoor" sk that allows us to open any v to any index x, by providing a value u such that $f_{pk,x}(u) = v$. For an adversary without this trapdoor, each output v would be computationally binding to x.

For an extractable HTDF, we would instead want $v = f_{pk,x}(u)$ to be statistically binding to x and to have an "extraction trapdoor" sk that allows us to extract/decrypt the value x from v. For an adversary without this trapdoor, the output v would computationally hide x.

In Section 3, we gave a construction of equivocable HTDFs with security based on the SIS problem. We now show that, by modifying the key generation procedure of our construction, we can easily convert it to an extractable HTDF. In fact, there is a single HTDF constriction with two *indistinguishable* modes of choosing the public key pk: in one mode, the resulting HTDF is equivocable with an equivocation trapdoor sk and in the other more the resulting HTDF is extractable with extraction trapdoor sk. The indistinguishability of these two modes follows from the learning with errors (LWE) assumption.

Recall that in our construction of HTDFs we set $pk = \mathbf{A}$ where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is (statistically close to) a uniformly random matrix. This corresponds to the equivocation mode. For the extractable mode we choose a matrix $\mathbf{A}' \leftarrow \mathbb{Z}_q^{(n-1)\times m}$ and a secret $\mathbf{s}' \leftarrow \mathbb{Z}_q^{n-1}$. We set

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}' \\ \mathbf{s}'\mathbf{A}' + \mathbf{e} \end{pmatrix}$$

where $\mathbf{e}$ is some appropriately sampled short "noise vector". This corresponds to a *learning with errors* (LWE) instance with secret $\mathbf{s}'$, and therefore $\mathbf{A}$ chosen as above is computationally indistinguishable from a uniformly random. Let $\mathbf{s} = (-\mathbf{s}', 1) \in \mathbb{Z}_q^n$ so that $\mathbf{s}\mathbf{A} = \mathbf{e}$. We set $pk = \mathbf{A}$

and $sk = \mathbf{s}$ to be the public key and secret extraction key of the HTDF. Otherwise, the construction $f_{pk,x}(\mathbf{U}) = \mathbf{A}\mathbf{U} + x\mathbf{G}$ and the homomorphic operations are performed the exact same way as in Section 3.2 and Section 3.3. Assume $\mathbf{V} = f_{pk,x}(\mathbf{U}) = \mathbf{A}\mathbf{U} + x\mathbf{G}$ where $\mathbf{U}$ is short. Let $\mathbf{z} = (0, \ldots, 0, r) \in \mathbb{Z}_q^n$ where r is a scalar of "medium size". We can then compute

$$\mathbf{s} \cdot \mathbf{V} \cdot \mathbf{G}^{-1}(\mathbf{z}) = \mathbf{e} \cdot \mathbf{U} \cdot \mathbf{G}^{-1}(\mathbf{z}) + x \cdot \langle \mathbf{s}, \mathbf{z} \rangle = x \cdot r + e'$$

where e' is short. As long as the parameters are chosen so that $|x \cdot r + e'| < q/2$ so there is no wrap-around, and $|e'| < |r|$, the above allows us to extract x.

With the above HTDF in extraction mode, if we think of $\mathsf{Enc}_{pk}(x; u) = f_{pk,x}(u)$ as a public-key encryption scheme with randomness u, and of the $\mathsf{SignEval}^{out}$ procedure as a homomorphic evaluation on ciphertexts, then this scheme corresponds to the FHE scheme of Gentry, Sahai and Waters [GSW13].