Leveled Strongly-Unforgeable Identity-Based Fully Homomorphic Signatures

Fuqun Wang[1,2,3(✉)], Kunpeng Wang[1,2], Bao Li[1,2], and Yuanyuan Gao[1,2,4]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{fqwang,kpwang,lb,yygao13}@is.ac.cn
[2] Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing, China
[3] University of Chinese Academy of Sciences, Beijing, China
[4] School of Science, Sichuan University of Science and Engineering, Zigong, China

Abstract. Recently, Gorbunov, Vaikuntanathan and Wichs proposed a new powerful primitive: (fully) homomorphic trapdoor function (HTDF) based on *small integer solution* (SIS) problem in standard lattices, from which they constructed the first leveled existentially-unforgeable fully homomorphic signature (FHS) schemes.

In this paper, we first extend the notion of HTDF to identity-based setting with stronger security and better parameters. The stronger security requires that the identity-based HTDF (IBHTDF) is not only *claw-free*, but also *collision-resistant*. And the maximum noise comparing to Gorbunov-Vaikuntanathan-Wichs' HTDF roughly reduces from $O(m^d\beta)$ to $O(4^d m\beta)$, which will result in polynomial modulus $q = \text{poly}(\lambda)$ when $d = O(\log \lambda)$, where λ is the security parameter and d is the depth bound of circuit. We then define and construct the first leveled strongly-unforgeable identity-based fully homomorphic signature (IBFHS) schemes.

Keywords: Identity-based homomorphic trapdoor function · Identity-based fully homomorphic signature · Small integer solution · Strong unforgeability

1 Introduction

Following the fast development of cloud computing, cryptographic schemes with homomorphic property attract a large number of researchers' sights. They allow a client to securely upload his/her encrypted/signed data to a remote server. Meanwhile they also allow the server to run computation over the data. The seminal study of fully homomorphic encryption (FHE) [17] demonstrates how

This work is supported in part by the National Nature Science Foundation of China under award No. 61272040 and No. 61379137, and in part by the National 973 Program of China under award No. 2013CB338001.

J. Lopez and C.J. Mitchell (Eds.): ISC 2015, LNCS 9290, pp. 42–60, 2015.
DOI: 10.1007/978-3-319-23318-5_3

to perform homomorphic computation over encrypted data without the knowledge of secret key. The recent works [6,18,23] of (leveled) fully homomorphic signatures demonstrate how to perform homomorphic computation on signed data.

In this work, we focus on the latter question: public authenticity of the result of homomorphic computation over signed data. In a homomorphic signature scheme, a client signs some data $\mathbf{x} = (x_1, \ldots, x_N)$ using his/her signing key and outsources the signed data $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_N)$ to a remote server. At any later point, the server can perform homomorphically some operation $y = g(\mathbf{x})$ over the signed data $\boldsymbol{\sigma}$ and produce a short signature σ_g certifying that y is the correct output of the operation g over the data $\mathbf{x}$. Anyone can verify the tuple (g, y, σ_g) using the client's public verification key and be sure of this fact without the knowledge of the underlying data $\mathbf{x}$.

Linear Homomorphic Signatures. A number of works discussed signatures with linear functions [2,4,10,16]. Such linear homomorphic signature schemes have meaningful applications in network coding and proofs of retrievability.

Somewhat Homomorphic Signatures. Boneh and Freeman [5] were the first to define and construct homomorphic signature schemes beyond linear functions, but limited to constant-degree polynomials based on ring SIS assumption in the random oracle model. Not long ago, Catalano, Fiore and Warinschi [11] gave an alternative scheme from multi-linear maps in the standard model.

Leveled Fully Homomorphic Signatures. Gorbunov, Vaikuntanathan and Wichs [18] proposed the first leveled FHS schemes based on SIS assumption. To this end, they drew on the ideas of constructing attribute-based encryption from standard lattices [7] and proposed a new primitive: HTDF. They required that HTDF functions have claw-freeness property, which is sufficient to show their FHS schemes (constructed directly from the HTDF functions) are existentially unforgeable in the static chosen-message-attack (EU-sCMA) model. Additionally, they showed that one can transform an EU-sCMA secure FHS to an EU-aCMA (existential-unforgeability under adaptive chosen-message-attack) secure FHS via homomorphic chameleon hash function. Recently, Boyen, Fan and Shi [6] also proposed EU-aCMA secure FHS schemes using vanishing trapdoor technique [1]. In the meantime, Xie and Xue [23] showed that leveled FHS schemes can be constructed if indistinguishability obfuscation and injective one way function exist.

1.1 Motivation

We observe that all schemes with homomorphism above are existentially unforgeable. In this model, a verifiable forgery (g, y', σ') such that g is admissible on messages $\mathbf{x}$ and $y' \neq y$ $(y = g(\mathbf{x}))$ captures two facts. One is that σ' is a usual existential-forgery corresponding to the usual notion of signature forgery if $g(\mathbf{x}) = \pi_i(\mathbf{x}) = x_i$ is a special projection function. The other is that σ' is a

homomorphic existential-forgery if g is a generally admissible function (defined in Sect. 3.1); in other words, the forgery σ' authenticates y' as $g(\mathbf{x})$ but in fact this is not the case.

However, as is well-known, security of signature schemes without homomorphism also can reach up to strong-unforgeability. In the stronger model, a forger can not give a forgery of message x_i, even he has a message-signature pair (x_i, σ_i). As a matter of course, we have a question: *can we define and construct strongly-unforgeable* (IB)FHS?

In this paper, we will give a positive response. Our main observation is that homomorphic computations on signed data are deterministic in all above schemes. In this scenario, we can define meaningful strong-unforgeability. In this model, given message-signature pairs $(\mathbf{x}, \boldsymbol{\sigma})$, a forger produce a verifiable strong-forgery (g, y', σ') such that $y' = y = g(\mathbf{x})$ and $\sigma' \neq \sigma_g$ that captures two facts. One is that $\sigma' \neq \sigma_i$ is a usual strong-forgery corresponding to the usual notion of strong-forgery if $g(\mathbf{x}) = \pi_i(\mathbf{x}) = x_i$. The other is that $\sigma' \neq \sigma_g$ is a homomorphic strong-forgery if g is a generally admissible function; in other words, the forgery σ' authenticates y' as $g(\mathbf{x})$ but in fact any forger can not produce $\sigma' \neq \sigma_g$.

Furthermore, as we all know, identity-based signature (IBS) is a nontrivial extension of signature [22]. In an IBS system, in order to verify a signature σ_i of a message x_i, the verifier requires only the global public parameters and the target identity *id*. Therefore, there is no need to issue a verification key for each user in an IBS system, which greatly simplifies the key management. Naturally, constructing an IBS with homomorphism is interesting. As far as we know, there is no construction of identity-based FHS. In fact, we will propose the first strongly-unforgeable IBFHS as a response to above question.

1.2 Contribution

We define and construct the first leveled strongly-unforgeable IBFHS schemes. To this end, we extend HTDF, the underlying primitive of FHS, to IBHTDF with stronger security and better parameters, the underlying primitive of IBFHS using the trapdoor technique in [1,12,19]. The stronger security requires that IBHTDFs are not only *claw-free*, but also *collision-resistant* to show the strong-unforgeability of IBFHS. We use Barrington's theorem to reduce the parameters as done in FHE world [9]. The maximum noise-level comparing to Gorbunov-Vaikuntanathan-Wichs' FHS roughly reduces from $O(m^d\beta)$ to $O(4^d m\beta)$, which will result in polynomial modulus $q = \text{poly}(\lambda)$ when $d = O(\log \lambda)$, where λ is the security parameter and d is the maximum depth of admissible circuit.

1.3 Paper Organization

In Sect. 2, we give some background on lattices and related tools as used in this paper. We propose formally the IBHTDF functions in Sect. 3 and demonstrate how to homomorphically evaluate a permutation branching program in Sect. 4. In Sect. 5, we define and construct the leveled strongly-unforgeable IBFHS. Finally, we conclude in Sect. 6.

2 Preliminaries

We use the bold upper-case letters (e.g., $\mathbf{A}, \mathbf{B}$) to represent matrices and bold lower-case letters (e.g. $\mathbf{a}, \mathbf{b}$) to represent column vectors. Let $\|\mathbf{A}\|_\infty = \max_{i,j}\{|a_{i,j}|\}$ denote the infinite norm and a_i or $\mathbf{a}[i]$ represent the i-entry of $\mathbf{a}$. Let $[\mathbf{A}||\mathbf{B}]$ denote the concatenation of two matrices and $(\mathbf{A}, \mathbf{B}) = [\mathbf{A}^T||\mathbf{B}^T]^T$. We use λ to denote the *security parameter* and $\mathrm{negl}(\lambda)$ to denote a negligible function that grows slower than λ^{-c} for any constant $c > 0$ and any large enough value of λ.

2.1 Entropy and Statistical Distance

For discrete random variables $X \leftarrow \mathcal{X}, Y \leftarrow \mathcal{Y}$, we define the *statistical distance* $\triangle(X,Y) \triangleq \frac{1}{2}\sum_{\omega\in\mathcal{X}\cup\mathcal{Y}} |\mathbf{Pr}[X = \omega] - \mathbf{Pr}[Y = \omega]|$. We say that two random variables X, Y are statistically indistinguishable, denoted as $X \approx_s Y$, if $\triangle(X,Y) = \mathrm{negl}(\lambda)$. The *min-entropy* of a random variable X, denoted by $\mathbf{H}_\infty(X)$, is defined as $\mathbf{H}_\infty(X) \triangleq -\log(\max_x \mathbf{Pr}[X = x])$. The *average min-entropy* of X conditioned on Y, denoted with $\widetilde{\mathbf{H}}_\infty(X|Y)$, is defined as

$$\widetilde{\mathbf{H}}_\infty(X|Y) \triangleq -\log\left(\mathbf{E}_{y\leftarrow\mathcal{Y}}[\max_x \mathbf{Pr}[X = x|Y = y]]\right) = -\log\left(\mathbf{E}_{y\leftarrow\mathcal{Y}}[2^{-\mathbf{H}_\infty(X|Y=y)}]\right).$$

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

Lemma 2.1 ([15]). *Let $X \leftarrow \mathcal{X}, Y \leftarrow \mathcal{Y}$ be two (correlated) random variables. It then holds that $\widetilde{\mathbf{H}}_\infty(X|Y) \geq \mathbf{H}_\infty(X) - \log(|\mathcal{Y}|)$.*

2.2 Background on Lattices and Hard Problems

Lattices. Lattices-based cryptography usually use so-called q-ary integer lattices, which contain $q\mathbb{Z}^m$ as a sublattice for some modulus q. Let n, m, q be positive integers. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ we define the following q-ary integer lattice:

$$\Lambda^\perp(\mathbf{A}) = \{\mathbf{u} \in \mathbb{Z}^m : \mathbf{A}\mathbf{u} = \mathbf{0} \mod q\}.$$

For a vector $\mathbf{v} \in \mathbb{Z}_q^n$, we define the coset (or "shifted" lattice):

$$\Lambda_\mathbf{v}^\perp(\mathbf{A}) = \{\mathbf{u} \in \mathbb{Z}^m : \mathbf{A}\mathbf{u} = \mathbf{v} \mod q\}.$$

SIS. Let n, m, q, β be integers. The short integer solution ($\mathrm{SIS}_{n,m,q,\beta}$) problem is, given a uniformly random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, to find a nonzero vector $\mathbf{u} \in \mathbb{Z}_q^n$ with $||\mathbf{u}||_\infty \leq \beta$ such that $\mathbf{A}\mathbf{u} = \mathbf{0}$ (i.e., $\mathbf{u} \in \Lambda^\perp(\mathbf{A})$). For $q \geq \beta \cdot \omega(\sqrt{n\log n})$, solving $\mathrm{SIS}_{n,m,q,\beta}$ in the average case is as hard as solving $\mathrm{GapSVP}_{\widetilde{O}(\beta\cdot\sqrt{n})}$ in the worst case in standard lattices [20,21].

Discrete Gaussian Distribution. Let $\mathcal{D}_{\mathbb{Z}^m,r}$ be the truncated discrete Gaussian distribution over $\mathbb{Z}^m$ with parameter r. Namely, for $\mathbf{u} \leftarrow \mathcal{D}_{\mathbb{Z}^m,r}$, if $\|\mathbf{u}\|_\infty$ is larger than $r \cdot \sqrt{m}$, then the output is replaced by $\mathbf{0}$. In other words, $\|\mathbf{u}\|_\infty \leq r \cdot \sqrt{m}$ with probability 1 if $\mathbf{u} \leftarrow \mathcal{D}_{\mathbb{Z}^m,r}$.

Lattices Trapdoor. Here we recall the MP12-trapdoor generation algorithm and Gaussian sampling algorithm [19]. We ignore all details of implementation which are not strictly necessary in this work.

For integers n, q and $\ell = \lceil \log q \rceil$, let $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^T \in \mathbb{Z}_q^{n \times n\ell}$, where $\mathbf{g}^T = (1, 2, 2^2, \ldots, 2^{\ell-1})$ and $\mathbf{I}_n$ denotes the n-dimensional identity matrix.

Lemma 2.2 ([19]). *Let n, q, ℓ, m_0, m_1 be integers such that $n = \mathrm{poly}(\lambda)$, $q = q(n), \ell = \lceil \log q \rceil, m_0 = n(\ell + O(1))$, $m_1 = n\ell$. For $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{n \times m_0}$ and $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$, there exists an randomized algorithm $\mathsf{TrapGen}(\mathbf{A}_0, \mathbf{H})$ to generate a matrix $\mathbf{A}$ $(= [\mathbf{A}_0 \| \mathbf{H}\mathbf{G} - \mathbf{A}_0\mathbf{R}]) \in \mathbb{Z}_q^{n \times (m_0+m_1)}$ with trapdoor $\mathbf{R}$ such that $\mathbf{R} \leftarrow \mathcal{D}_{\mathbb{Z}^{m_0 \times m_1},r}$ for large enough r $(\geq \omega(\sqrt{\log n}))$ and $\mathbf{A}$ is $\mathrm{negl}(\lambda)$-far from $(\mathbf{V}_0, \mathbf{V}_1) \xleftarrow{\$} \mathbb{Z}_q^{n \times m_0} \times \mathbb{Z}_q^{n \times m_1}$. Here, $\mathbf{R}$ is called an MP12-trapdoor (or $\mathbf{G}$-trapdoor) of $\mathbf{A}$ with tag $\mathbf{H}$.*

Furthermore, for any non-zero $\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1) \in \mathbb{Z}_q^{m_0+m_1}$, the average min-entropy of $\mathbf{R}\mathbf{u}_1$ given $\mathbf{A}_0$ and $\mathbf{A}_0\mathbf{R}$ is at least $\Omega(n)$.

Lemma 2.3 ([19]). *Given parameters in above lemma and a uniformly random vector $\mathbf{v} \in \mathbb{Z}_q^n$, for some s $(\geq O(\sqrt{n \log q})) \in \mathbb{R}$ and a fixed function $\omega(\sqrt{\log n})$ growing asymptotically faster than $\sqrt{\log n}$, if the tag matrix $\mathbf{H}$ is invertible, there then exists an efficient algorithm $\mathsf{SamplePre}(\mathbf{A}_0, \mathbf{R}, \mathbf{H}, \mathbf{v}, s)$ that samples a vector $\mathbf{u}$ from $\mathcal{D}_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{A}), s \cdot \omega(\sqrt{\log n})}$ such that $\mathbf{A} \cdot \mathbf{u} = \mathbf{v}$. Note that $\|\mathbf{u}\|_\infty \leq s\sqrt{m_0 + m_1} \cdot \omega(\sqrt{\log n})$ with probability 1.*

Furthermore, for $\mathbf{u}' \leftarrow \mathcal{D}_{\mathbb{Z}^m, s \cdot \omega(\sqrt{\log n})}$ and $\mathbf{v}' = \mathbf{A}\mathbf{u}'$, we have $(\mathbf{A}, \mathbf{R}, \mathbf{u}, \mathbf{v}) \approx_s (\mathbf{A}, \mathbf{R}, \mathbf{u}', \mathbf{v}')$.

Lemma 2.4 ([7,18,19]). *Let $m = m_0 + 2m_1$ and $\widetilde{\mathbf{G}} = [\mathbf{G} \| \mathbf{0}] \in \mathbb{Z}_q^{n \times m}$. For any matrix $\mathbf{V} \in \mathbb{Z}_q^{n \times m}$ there exists deterministic algorithm to output a $\{0,1\}$-matrix $\widehat{\mathbf{V}} \in \mathbb{Z}_q^{m \times m}$ such that $\widetilde{\mathbf{G}}\widehat{\mathbf{V}} = \mathbf{V}$ (or denoted by $\widetilde{\mathbf{G}}^{-1}(\mathbf{V}) = \widehat{\mathbf{V}}$[1]).*

2.3 Permutation Branching Program.

In this section, we define permutation branching program closely following [9]. A width-w permutation branching program Π of length L with input space $\{0,1\}^t$ is a sequence of L tuples of the form $(h(k), \sigma_{k,0}, \sigma_{k,1})$ where

- $h : [L] \to [t]$ is a function associates the k-th tuple with an input bit $x_{h(k)}$.
- $\sigma_{k,0}, \sigma_{k,1}$ are permutations over $[w] = \{1, 2, \ldots, w\}$.

[1] Here $\widetilde{\mathbf{G}}^{-1}$ is not the inverse matrix of $\widetilde{\mathbf{G}}$ but a deterministic algorithm.

A permutation branching program Π performs evaluation on input $\mathbf{x} = (x_1, x_2, \ldots, x_t)$ as follows. Let the initial state be $\eta_0 = 1$ and the k-th state be $\eta_k \in [w]$. We compute the state η_k recursively as

$$\eta_k = \sigma_{k,x_{h(k)}}(\eta_{k-1}).$$

Finally, after L steps, the end state is η_L. The output of Π is 1 if $\eta_L = 1$, and 0 otherwise.

To slow the growth of noise in homomorphic operations, we represent the states to bits, as demonstated in [9]. More specially, we replace the state $\eta_k \in [w]$ with some w-dimensional unit vector $\mathbf{v}_k$, e.g., $\mathbf{v}_0 = (1, 0, 0, \ldots, 0)$ institutes for $\eta_0 = 1$. The idea is that $\mathbf{v}_k[i] = 1$ if and only if $\sigma_{k,x_{h(k)}}(\eta_{k-1}) = i$. A more important equivalent relation is that $\mathbf{v}_k[i] = 1$ if and only if either:

- $x_{h(k)} = 1$ and $\mathbf{v}_{k-1}[\sigma_{k,1}^{-1}(i)] = 1$; or
- $x_{h(k)} = 0$ and $\mathbf{v}_{k-1}[\sigma_{k,0}^{-1}(i)] = 1$.

Hence, for $k \in [L], i \in [w]$, we have

$$\begin{aligned}\mathbf{v}_k[i] &= \mathbf{v}_{k-1}[\sigma_{k,1}^{-1}(i)] \cdot x_{h(k)} + \mathbf{v}_{k-1}[\sigma_{k,0}^{-1}(i)] \cdot (1 - x_{h(k)}) \\ &= \mathbf{v}_{k-1}[\gamma_{k,i,1}] \cdot x_{h(k)} + \mathbf{v}_{k-1}[\gamma_{k,i,0}] \cdot (1 - x_{h(k)}) \end{aligned} \quad (1)$$

where $\gamma_{k,i,1} \triangleq \sigma_{k,1}^{-1}(i)$ and $\gamma_{k,i,0} \triangleq \sigma_{k,0}^{-1}(i)$ are fully determined by the description of Π and can be computed easily and publicly. Thus, $\{(h(k), \gamma_{k,i,0}, \gamma_{k,i,1})\}_{k\in[L],i\in[w]}$ is an alternative description of a permutation branching program and is the form that we will work with under homomorphic computations.

3 Identity-Based Homomorphic Trapdoor Functions

We give the definition, construction and security proof of IBHTDFs in this section. In next section we will show how to homomorphically compute a circuit. Looking ahead, we will homomorphically compute a permutation branching program instead of a (boolean) circuit to reduce the parameters and increase the efficiency and security.

3.1 Definition

An *identity-based homomorphic trapdoor function* (IBHTDF) consists of six poly-time algorithms ($\mathsf{IBHTDF.Setup}, \mathsf{IBHTDF.Extract}, f, \mathsf{Invert}, \mathsf{IBHTDF.Eval}^{in}, \mathsf{IBHTDF.Eval}^{out}$) with syntax as follows:

- $(mpk, msk) \leftarrow \mathsf{IBHTDF.Setup}(1^\lambda)$: A master key setup procedure.
  The security parameter λ defines the identity space $\mathcal{I}$, the index space $\mathcal{X}$, the input space $\mathcal{U}$, the output space $\mathcal{V}$ and some efficiently samplable input distribution $\mathcal{D}_\mathcal{U}$ over $\mathcal{U}$. We require that elements in $\mathcal{I}, \mathcal{U}, \mathcal{V}$ or $\mathcal{X}$ can be efficiently certified and that one can efficiently sample elements from $\mathcal{V}$ uniformly at random.

- $(pk_{id}, sk_{id}) \leftarrow \mathsf{IBHTDF.Extract}(mpk, msk, id)$: An identity-key extraction procedure. As a matter of course, we require that pk_{id} can be extracted deterministically from mpk and $id \in \mathcal{I}$ without using the knowledge of msk.
- $f_{pk_{id},x} : \mathcal{U} \rightarrow \mathcal{V}$: A deterministic function indexed by pk_{id} and $x \in \mathcal{X}$.
- $\mathsf{Invert}_{sk_{id},x} : \mathcal{V} \rightarrow \mathcal{U}$: A probabilistic inverter indexed by sk_{id} and $x \in \mathcal{X}$.
- $u_g = \mathsf{IBHTDF.Eval}^{in}(g, (x_1, u_1, v_1), \ldots, (x_t, u_t, v_t))$: A deterministic *input* homomorphic evaluation algorithm. It takes as input some function $g : \mathcal{X}^t \rightarrow \mathcal{X}$ and values $\{x_i \in \mathcal{X}, u_i \in \mathcal{U}, v_i \in \mathcal{V}\}_{i\in[t]}$ and outputs $u_g \in \mathcal{U}$.
- $v_g = \mathsf{IBHTDF.Eval}^{out}(g, v_1, \ldots, v_t)$: A deterministic *output* homomorphic evaluation algorithm. It takes as input some function $g : \mathcal{X}^t \rightarrow \mathcal{X}$ and values $\{v_i \in \mathcal{V}\}_{i\in[t]}$ and outputs $v_g \in \mathcal{V}$.

Correctness of Homomorphic Computation. Let algorithm $(pk_{id}, sk_{id}) \leftarrow \mathsf{IBHTDF.Extract}$ extracts the identity-key for id. Let $g : \mathcal{X}^t \rightarrow \mathcal{X}$ be a function on $x_1, \ldots, x_t \in \mathcal{X}$ and set $y = g(x_1, \ldots, x_t)$. Let $u_1, \ldots, u_t \in \mathcal{U}$ and set $v_i = f_{pk_{id},x}(u_i)$ for $i = 1, \ldots, t$. Set $u_g = \mathsf{IBHTDF.Eval}^{in}(g, (x_1, u_1, v_1), \ldots, (x_t, u_t, v_t))$, $v_g = \mathsf{IBHTDF.Eval}^{out}(g, v_1, \ldots, v_t)$. We require that $u_g \in \mathcal{U}$ and $f_{pk_{id},x}(u_g) = v_g$.

Relaxation Correctness of Leveled IBHTDFs. In a leveled IBHTDF, every input $u_i \in \mathcal{U}$ will carry with noise $\beta_i \in \mathbb{Z}$. The initial samples chosen from the input-distribution $\mathcal{D}_{\mathcal{U}}$ carry with small noise β_0 and the noise β_g of the homomorphically evaulation u_g depends on the noise β_i of u_i, the indices x_i and the function g. In fact, if the noise $\beta_g > \beta_{max}$, where β_{max} is a threshold of noise, there is no guarantee of the correctness. Therefore, we should restrict the class of functions that can be computed. We say a function g is *admissible* on indices $x_1, \ldots, x_t$ if $\beta_g \leq \beta_{max}$ whenever u_i carries with noise $\beta_i \leq \beta_0$.

Distributional Equivalence of Inversion. To show the security of our main construction IBFHS in next section, we require the following statistical indistinguishability:

$$(pk_{id}, sk_{id}, x, u, v) \approx_s (pk_{id}, sk_{id}, x, u', v')$$

where $(pk_{id}, sk_{id}) \leftarrow \mathsf{IBHTDF.Extract}$, $x \in \mathcal{X}$, $u \leftarrow \mathcal{D}_{\mathcal{U}}, v = f_{pk_{id},x}(u), v' \xleftarrow{\$} \mathcal{V}, u' \leftarrow \mathsf{Invert}_{sk_{id},x}(v')$.

IBHTDF Security. Gorbunov *et al.* [18] required *claw-freeness* for HTDF security to provide *existential-unforgeability* for FHS. Here, we require not only *claw-freeness* but also *collision-resistance* for IBHTDF security to guarantee *strong-unforgeability* for IBFHS.

The experiment $\mathbf{Exp}^{\mathrm{sID}}_{\mathcal{A},\mathsf{IBHTDF}}(1^\lambda)$ defined in Fig. 1 describes the *selective-identity* security, where the adversary has to appoint a target identity id^* to attack before seeing the master public-key. Moreover, the adversary can query identity-keys for all identities except id^*. He is then forced to find $u \neq u' \in \mathcal{U}$, $x, x' \in \mathcal{X}$ such that $f_{pk_{id^*},x}(u) = f_{pk_{id^*},x'}(u')$. Remark that if $x = x'$, then (u, u') is a collision, a claw otherwise.

$$\mathbf{Exp}^{\mathrm{sID}}_{\mathcal{A},\mathsf{IBHTDF}}(1^\lambda)$$

- $(id^*, state) \leftarrow \mathcal{A}(1^\lambda)$
- $(mpk, msk) \leftarrow \mathsf{IBHTDF.Setup}(1^\lambda)$
- $(u, u', x, x') \leftarrow \mathcal{A}^{\mathsf{IBHTDF.Extract}(mpk,msk,\cdot)\setminus\{id^*\}}(mpk, state)$
- $\mathcal{A}$ wins if $u \neq u' \in \mathcal{U},\ x, x' \in \mathcal{X}$ are such that $f_{pk_{id^*},x}(u) = f_{pk_{id^*},x'}(u')$.

Fig. 1. Definition of *selective-identity* security for IBHTDF

We say that an identity-based homomorphic trapdoor function is *selective-identity* secure if $\Pr[\mathbf{Exp}^{\mathrm{sID}}_{\mathcal{A},\mathsf{IBHTDF}}(1^\lambda)] \leq \mathrm{negl}(\lambda)$.

In the stronger model of *adaptive-identity security*, the adversary can not find $u \neq u' \in \mathcal{U},\ x, x' \in \mathcal{X}$ such that $f_{pk_{id},x}(u) = f_{pk_{id},x'}(u')$ for any identity id, for which he has never queried identity-key sk_{id}. We note that one may construct *adaptive-identity* secure IBHTDF using the *vanishing trapdoor* techniques [1,8, 12] in the cost of both efficiency and security.

3.2 Construction: Basic Algorithms and Security

Recall that λ is the security parameter. To describe the IBHTDF functions succinctly, we give some public parameters as follows.

- Let flexible d be the circuit depth such that $d \leq \mathrm{poly}(\lambda)$ and set $L = 4^d$.
- Choose an integer $n = \mathrm{poly}(\lambda)$ and a sufficiently large prime $q = q(n)$. Let $\ell = \lceil \log q \rceil$, $m_0 = n(\ell + O(1))$, $m_1 = n\ell$ and $m = m_0 + 2m_1$. Set $\beta_0 = O((n \log q)^{3/2}), \beta_{max} = O(4^d m \beta_0), \beta_{SIS} = O(m_1 \beta_0)\beta_{max} < q$.
- $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^T \in \mathbb{Z}_q^{n \times n\ell}$ is the primitive matrix, where $\mathbf{g}^T = (1, 2, 2^2, \ldots, 2^{\ell-1})$. Set $\widetilde{\mathbf{G}} = [\mathbf{G} \| \mathbf{0}] \in \mathbb{Z}_q^{n \times m}$ be the garget matrix used below.
- We assume that identities are elements in $\mathrm{GF}(q^n)$, and say $\mathbf{H} : \mathrm{GF}(q^n) \to \mathbb{Z}_q^{n \times n}$ is an invertible difference, if $\mathbf{H}(id_1) - \mathbf{H}(id_2)$ is invertible for any two different identities id_1, id_2 and $\mathbf{H}$ is computable in polynomial time in $n\ell$ (see an example in [1]).
- Set $\mathcal{X} = \mathbb{Z}_2, \mathcal{I} = \mathbb{Z}_q^n, \mathcal{V} = \mathbb{Z}_q^{n \times m}$ and $\mathcal{U} = \{\mathbf{U} \in \mathbb{Z}_q^{m \times m} : \|\mathbf{U}\|_\infty \leq \beta_{max}\}$. Define the distribution $\mathcal{D}_\mathcal{U}$ is a truncated discrete Gaussian distribution over $\mathcal{U}$, so that $\|\mathbf{U}\|_\infty \leq \beta_0$ if $\mathbf{U} \leftarrow \mathcal{D}_\mathcal{U}$.

Now we describe the basic algorithms of IBHTDF function $\mathcal{F}$.

- $\mathsf{IBHTDF.Setup}(1^\lambda)$: On input a security parameter λ, set $d, L, n, m_0, m_1, m, q, \beta_0, \beta_{max}, \beta_{SIS}$ as specified above. Then do:
  1. Choose $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{n \times m_0}$. Run $\mathsf{TrapGen}(\mathbf{A}_0, \mathbf{0})$ to generate a matrix $\mathbf{A} = [\mathbf{A}_0 || \mathbf{A}_1] = [\mathbf{A}_0 || -\mathbf{A}_0\mathbf{R}] \in \mathbb{Z}_q^{n \times (m_0 + m_1)}$ and a trapdoor $\mathbf{R}$ such that $\mathbf{R} \leftarrow \mathcal{D} \triangleq \mathcal{D}_{\mathbb{Z}^{m_0 \times m_1}, \omega(\sqrt{\log n})}$ and $\mathbf{A}$ is $\mathrm{negl}(\lambda)$-far from uniform. Set the master secret key as $msk = \mathbf{R}$. Note that $\mathbf{A} \cdot (\mathbf{R}, \mathbf{I}_{m_1}) = \mathbf{0}$, namely $\mathbf{R}$ is a $\mathbf{G}$-trapdoor of $\mathbf{A}$ with tag $\mathbf{0}$.

2. Choose $\mathbf{A}_2 \xleftarrow{\$} \mathbb{Z}_q^{n\times m_1}$ and set the master public key as $mpk = \{\mathbf{A}, \mathbf{A}_2\}$.

– $\mathsf{IBHTDF.Extract}(mpk, \mathbf{R}, id)$: On input a master public key mpk, a master secret key $\mathbf{R}$ and an identity $id \in \mathcal{I}$, do:
1. Compute $\mathbf{H}(id)$ for $id \in \mathcal{I}$ and let $\mathbf{A}'_{id} = [\mathbf{A}_0||\mathbf{H}(id)\cdot\mathbf{G} + \mathbf{A}_1]$ (Note that $\mathbf{R}$ is a $\mathbf{G}$-trapdoor of $\mathbf{A}'_{id}$ with tag $\mathbf{H}(id)$). Set user-specific public-key $pk_{id} = \mathbf{A}_{id} = [\mathbf{A}'_{id}||\mathbf{A}_2]$.
2. Run algorithm $\mathsf{SamplePre}(\mathbf{A}_0, \mathbf{R}, \mathbf{H}(id), \mathbf{G} - \mathbf{A_2}, O(\sqrt{n\log q}))$ to output $\mathbf{R}_{id} \in \mathbb{Z}^{(m_0+m_1)\times m_1}$ such that $\mathbf{A}'_{id}\cdot\mathbf{R}_{id} = \mathbf{G} - \mathbf{A}_2$ (Note that $\mathbf{R}_{id}$ is a $\mathbf{G}$-trapdoor of $\mathbf{A}_{id}$ with tag $\mathbf{I}_n$). Set secret key $sk_{id} = \mathbf{R}_{id}$.

– $f_{pk_{id},x}(\mathbf{U})$: On input $mpk, id \in \mathcal{I}, x \in \mathcal{X}$ and $\mathbf{U} \in \mathcal{U}$, do:
1. Compute $pk_{id} = \mathbf{A}_{id} = [\mathbf{A}_0||\mathbf{H}(id)\cdot\mathbf{G} + \mathbf{A}_1||\mathbf{A}_2]$ as above.
2. For $id \in \mathcal{I}, x \in \mathcal{X}$ and $\mathbf{U} \in \mathcal{U}$, define $f_{pk_{id},x}(\mathbf{U}) \triangleq \mathbf{A}_{id}\cdot\mathbf{U} + x\cdot\widetilde{\mathbf{G}}$.

– $\mathsf{Invert}_{sk_{id},x}(\mathbf{V})$: On input an identity $id \in \mathcal{I}$, an identity-key $\mathbf{R}_{id}$, an index $x \in \mathcal{X}$ and $\mathbf{V} \in \mathcal{V}$, run $\mathsf{SamplePre}(\mathbf{A}'_{id}, \mathbf{R}_{id}, \mathbf{I}_n, \mathbf{V} - x\cdot\widetilde{\mathbf{G}}, O(n\log q))$ to output $\mathbf{U}$ (such that $\mathbf{A}_{id}\cdot\mathbf{U} = \mathbf{V} - x\cdot\widetilde{\mathbf{G}}$).

Distributional Equivalence of Inversion. Let $x \in \mathcal{X}$ and $(pk_{id} = \mathbf{A}_{id}, sk_{id} = \mathbf{R}_{id}) \leftarrow \mathsf{IBHTDF.Extract}(mpk, \mathbf{R}, id)$. Let $\mathbf{U} \in \mathcal{U}$, $\mathbf{V} = f_{pk_{id},x}(\mathbf{U}) = \mathbf{A}_{id}\cdot\mathbf{U} + x\widetilde{\mathbf{G}}$, $\mathbf{V}' \xleftarrow{\$} \mathcal{V}$, $\mathbf{U}' \leftarrow \mathsf{SamplePre}(\mathbf{A}'_{id}, \mathbf{R}_{id}, \mathbf{I}_n, \mathbf{V}' - x\widetilde{\mathbf{G}}, O(n\log q))$. By Lemma 2.3 and the fact that $(\mathbf{V}' - x\widetilde{\mathbf{G}})$ is uniformly random, using a simple hybrid argument, we have

$$(\mathbf{A}_{id}, \mathbf{R}_{id}, \mathbf{U}, \mathbf{A}_{id}\cdot\mathbf{U}) \approx_s (\mathbf{A}_{id}, \mathbf{R}_{id}, \mathbf{U}', \mathbf{V}' - x\widetilde{\mathbf{G}}).$$

Then, we have

$$(\mathbf{A}_{id}, \mathbf{R}_{id}, x, \mathbf{U}, \mathbf{V} = \mathbf{A}_{id}\cdot\mathbf{U} + x\widetilde{\mathbf{G}}) \approx_s (\mathbf{A}_{id}, \mathbf{R}_{id}, x, \mathbf{U}', \mathbf{V}') \tag{2}$$

by applying the same function to both sides: put in a $x \in \mathcal{X}$ and add $x\widetilde{\mathbf{G}}$ to the last entry.

IBHTDF Security. We now show that the IBHTDF function $\mathcal{F}$ constructed above is selective-identity secure assuming the SIS assumption.

Theorem 3.1. *The function $\mathcal{F}$ constructed above is a selective-identity secure* IBHTDF *assuming the* $\mathrm{SIS}_{n,m_0,q,\beta_{SIS}}$ *assumption.*

Proof. Assume there exists a PPT adversary $\mathcal{A}$ that wins the security experiment $\mathbf{Exp}^{\mathrm{sID}}_{\mathcal{A},\mathsf{IBHTDF}}(1^\lambda)$ for $\mathcal{F}$ with non-negligible probability δ. We construct a PPT simulater $\mathcal{S}$ that breaks the $\mathrm{SIS}_{n,m_0,q,\beta_{SIS}}$ problem for $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{n\times m_0}$.

Let id^* be the identity that $\mathcal{A}$ intends to attack. $\mathcal{S}$ will run the simulated algorithms $(\mathsf{IBHTDF.Setup}^*, \mathsf{IBHTDF.Extract}^*)$.

– $\mathsf{IBHTDF.Setup}^*(1^\lambda)$: On input the same parameters as $\mathsf{IBHTDF.Setup}(1^\lambda)$, $\mathcal{S}$ does:

1. After receiving target identity $id^* \in \mathcal{I}$ and challenge matrix $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m_0}$, $\mathcal{S}$ runs $\mathsf{TrapGen}(\mathbf{A}_0, -\mathbf{H}(id^*))$ to produce a matrix $\mathbf{A} = [\mathbf{A}_0||\mathbf{A}_1] = [\mathbf{A}_0|| - \mathbf{H}(id^*)\mathbf{G} - \mathbf{A}_0\mathbf{R}] \in \mathbb{Z}_q^{n \times (m_0+m_1)}$ and a trapdoor $\mathbf{R}$ such that $\mathbf{R} \leftarrow \mathcal{D}$ and $\mathbf{A}$ is negl(λ)-far from uniform. Set $msk = \mathbf{R}$.
2. $\mathcal{S}$ samples $\mathbf{S} \leftarrow \mathcal{D}$ and computes $\mathbf{A}_2 = \mathbf{A}_0\mathbf{S}$. Set $mpk = \{\mathbf{A}, \mathbf{A}_2\}$.

– $\mathsf{IBHTDF.Extract}^*(mpk, \mathbf{R}, id)$: On input a master public key mpk, a master secret key $\mathbf{R}$ and an identity $id \in \mathcal{I}$, do:

1. Compute $\mathbf{H}(id)$ for $id \in \mathcal{I}$ and let $\mathbf{A}'_{id} = [\mathbf{A}_0||\mathbf{H}(id) \cdot \mathbf{G} + \mathbf{A}_1] = [\mathbf{A}_0||(\mathbf{H}(id) - \mathbf{H}(id^*))\mathbf{G} - \mathbf{A}_0\mathbf{R}]$ (Note that $\mathbf{R}$ is a $\mathbf{G}$-trapdoor of $\mathbf{A}'_{id}$ with tag $\mathbf{H}(id) - \mathbf{H}(id^*)$. Set $\mathbf{A}_{id} = [\mathbf{A}'_{id}||\mathbf{A}_2]$.
2. Recall that $(\mathbf{H}(id) - \mathbf{H}(id^*))$ is invertible (by the property of $\mathbf{H}$) if $id \neq id^*$. Therefore, to respond to an identity-key query for $id \neq id^*$, $\mathcal{S}$ can run $\mathsf{SamplePre}(\mathbf{A}_0, \mathbf{R}, \mathbf{H}(id) - \mathbf{H}(id^*), \mathbf{G} - \mathbf{A_2}, O(\sqrt{n \log q}))$ and output $\mathbf{R}_{id} \in \mathbb{Z}^{(m_0+m_1) \times m_1}$ such that $\mathbf{A}'_{id} \cdot \mathbf{R}_{id} = \mathbf{G} - \mathbf{A}_2$ (Note that $\mathbf{R}_{id}$ is a $\mathbf{G}$-trapdoor of $\mathbf{A}_{id}$ with tag $\mathbf{I}_n$). Set $pk_{id} = \mathbf{A}_{id}$ and $sk_{id} = \mathbf{R}_{id}$.
3. However, if $id = id^*$, then $\mathbf{A}_{id^*} = [\mathbf{A}_0|| - \mathbf{A}_0\mathbf{R}||\mathbf{A}_0\mathbf{S}]$ and the trapdoor disappears. Thus, the simulator $\mathcal{S}$ can not generate identity key for id^*.

The views of adversary $\mathcal{A}$ between the original experiment and the simulated experiment are indistinguishable by Lemma 2.2. Particularly, the winning probability of $\mathcal{A}$ attacking the simulated experiment is at least $\delta - \text{negl}(\lambda)$.

Now, we show that an adversary $\mathcal{A}$ who wins the simulated experiment $\mathbf{Exp}^{\text{sID}}_{\mathcal{A},\mathsf{IBHTDF}}(1^\lambda)$ can be used to solve the SIS problem. Assume the winning adversary $\mathcal{A}$ outputs values $\mathbf{U} \neq \mathbf{U}' \in \mathcal{U}$, $x, x' \in \mathcal{X}$ such that $f_{pk_{id^*},x}(\mathbf{U}) = f_{pk_{id^*},x'}(\mathbf{U}')$. Let $\mathbf{U}^* = \mathbf{U} - \mathbf{U}'$ and $x^* = x' - x$. Then,

$$f_{pk_{id^*},x}(\mathbf{U}) = \mathbf{A}_{id^*}\mathbf{U} + x\widetilde{\mathbf{G}} = \mathbf{A}_{id^*}\mathbf{U}' + x'\widetilde{\mathbf{G}} = f_{pk_{id^*},x'}(\mathbf{U}') \;\Rightarrow\; \mathbf{A}_{id^*}\mathbf{U}^* = x^*\widetilde{\mathbf{G}}. \tag{3}$$

Recall that $\mathbf{A}_{id^*} = [\mathbf{A}_0|| - \mathbf{A}_0\mathbf{R}||\mathbf{A}_0\mathbf{S}]$. By the right hand side of Eq. (3), it holds that

$$\mathbf{A}_0 \cdot \mathbf{U}^\diamond \triangleq \mathbf{A}_0 \cdot ([\mathbf{I}_{m_0}|| - \mathbf{R}||\mathbf{S}]\mathbf{U}^*) = x^*\widetilde{\mathbf{G}}. \tag{4}$$

Moreover, since $\mathbf{U}, \mathbf{U}' \in \mathcal{U}$, we have $\|\mathbf{U}\|_\infty, \|\mathbf{U}'\|_\infty \leq \beta_{max}$ and thus $\|\mathbf{U}^*\|_\infty \leq 2\beta_{max}$. Moveover, since $\mathbf{R}, \mathbf{S}$ are sampled from $\mathcal{D}$, we also have $\|\mathbf{R}\|_\infty$, $\|\mathbf{S}\|_\infty \leq O(\sqrt{n \log q})$ and thus $\|\mathbf{U}^\diamond\|_\infty \leq 2\beta_{max}(2m_1 \cdot O(\sqrt{n \log q}) + 1) \leq \beta_{SIS}$.

To solve the SIS problem defined by $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m_0}$, we discuss the following two cases:

– $x = x'$ (collision): In this case, it is sufficed to show that $\mathbf{U}^\diamond \neq 0$ except with negligible probability, since $\mathbf{A}_0\mathbf{U}^\diamond = x^*\widetilde{\mathbf{G}} = (x - x')\widetilde{\mathbf{G}} = 0$ and $\|\mathbf{U}^\diamond\|_\infty$ is small. Let $\mathbf{U}^* = (\mathbf{U}_0^*, \mathbf{U}_1^*, \mathbf{U}_2^*)$. Then, we have $\mathbf{U}^\diamond = \mathbf{U}_0^* - \mathbf{R}\mathbf{U}_1^* + \mathbf{S}\mathbf{U}_2^*$. We split to 2 distinct cases to analyze it.

1. $\mathbf{U}_1^* = \mathbf{U}_2^* = 0$: In this case, we have $\mathbf{U}_0^* \neq 0$ since $\mathbf{U}^* \neq 0$. So, $\mathbf{U}^\diamond \neq 0$.
2. $\mathbf{U}_1^* \neq 0$ or $\mathbf{U}_2^* \neq 0$: Without loss of generalization, we assume $\mathbf{U}_2^* \neq 0$. By Lemma 2.2, we then have that, even revealing $\mathbf{R}$, the min-entropy of $\mathbf{S}\mathbf{U}_2^*$ conditioned on the knowledge of $\mathbf{A}_0$ and $\mathbf{A}_0\mathbf{S}$ is at least $\Omega(n)$. Particularly, the probability that $\mathbf{U}^\diamond = \mathbf{0}$ is less than $2^{-\Omega(n)} = \text{negl}(\lambda)$.

– $x \neq x'$ (claw): In this case, we show that the simulater $\mathcal{S}$ can use the knowledge of a small $\mathbf{U}^{\diamond} \neq \mathbf{0}$ and some $x^* \neq 0$ satisfying the Eq. (4) to find a solution of the SIS problem (similarly as [18]).
Choose $\mathbf{t} \xleftarrow{\$} \{0,1\}^{m_0}$ and set $\mathbf{r} \triangleq \mathbf{A}_0\mathbf{t}$. Compute $\mathbf{t}' = \widetilde{\mathbf{G}}^{-1}(\mathbf{r}/x^*) \in \{0,1\}^m$ such that $x^*\widetilde{\mathbf{G}}\mathbf{t}' = \mathbf{r}$. so,

$$\mathbf{A}_0(\mathbf{U}^{\diamond}\mathbf{t}' - \mathbf{t}) = (\mathbf{A}_0\mathbf{U}^{\diamond})\mathbf{t}' - \mathbf{A}_0\mathbf{t} = x^*\widetilde{\mathbf{G}}\mathbf{t}' - \mathbf{A}_0\mathbf{t} = \mathbf{r} - \mathbf{r} = 0.$$

Setting $\mathbf{u} \triangleq \mathbf{U}^{\diamond}\mathbf{t}' - \mathbf{t}$, we then have $\mathbf{A}_0\mathbf{u} = 0$ and $\|\mathbf{u}\|_\infty \leq (2m+1)\beta_{max} \leq \beta_{SIS}$. It remains to prove that $\mathbf{u} \neq 0$, i.e., $\mathbf{t} \neq \mathbf{U}^{\diamond}\mathbf{t}'$. We prove that it holds with overwhelming probability over the random $\mathbf{t}$, even given $\mathbf{A}_0, \mathbf{U}^{\diamond}, x^*$. In fact, we have

$$\widetilde{\mathbf{H}}_\infty(\mathbf{t}|\mathbf{t}') \geq \widetilde{\mathbf{H}}_\infty(\mathbf{t}|\mathbf{A}_0\mathbf{t}) \geq m_0 - n\log q = O(n).$$

where the first inequality follows from the fact that $\mathbf{t}'$ is deterministic by $\mathbf{r} = \mathbf{A}_0\mathbf{t}$, and the second inequality follows from Lemma 2.1. So, $\Pr[\mathbf{t} = \mathbf{U}^{\diamond}\mathbf{t}'] \leq 2^{-O(n)} = \mathrm{negl}(\lambda)$.

Therefore, if the adversary $\mathcal{A}$ wins the simulated experiment $\mathbf{Exp}^{\mathrm{sID}}_{\mathcal{A},\mathsf{IBHTDF}}(1^\lambda)$ with non-negligible probability $\delta/2 - \mathrm{negl}(\lambda)$ in either case, the simulater $\mathcal{S}$ then will produce a valid solution for SIS problem with probability $\delta/2 - \mathrm{negl}(\lambda)$. This finishes the proof. □

4 Homomorphic Evaluation and Noise Analysis

Although we can homomorphically compute arithmetic circuit or boolean circuit similarly as that in [18] with same-level parameters, we show how to do better in both works in this section based on the fact that the noise growth is asymmetric.

We define deterministic homomorphic addition and multiplication algorithms in Sect. 4.1. In Sect. 4.2, we show that these algorithms are not used by a naive combination of addition and multiplication, as in the work [9], but by an elaborate combination form to considerably slowing down the noise growth. The main difference between this work and [9] is that, to homomorphic evaluate, it requires us to design correspondingly two deterministic homomorphic algorithms: one for *input* and the other for *output* in this work, while it only requires to design one randomized homomorphic algorithm over ciphertexts in [9].

4.1 Basic Homomorphic Evaluation

We now define basic homomorphic addition and multiplication algorithms that will be used in IBHTDFs. These algorithms for IBHTDFs are same as that for HTDFs in [18] because of the same external structure with or without identity. Therefore, we can improve the parameters of HTDFs in [18] using asymmetric homomorphic multiplication demonstrated in this section and simplify the notations (e.g., Add^{in} instead of $\mathsf{IBHTDF.Add}^{in}$). Recall that $\mathbf{V}_i = \mathbf{A}\mathbf{U}_i + x_i\mathbf{G}$ $(i = 1, 2)$, where we set $\mathbf{A} = \mathbf{A}_{id}, \mathbf{G} = \widetilde{\mathbf{G}}$ for simplicity throughout Sect. 4. Let $\|\mathbf{U}_i\|_\infty \leq \beta_i$ and $x_i \in \{0,1\}$.

Homomorphic Addition Algorithms. They are simple modulo-q addition of the *input* or *output* matrices respectively.

- $\mathsf{Add}^{in}((x_1, \mathbf{U}_1, \mathbf{V}_1), (x_2, \mathbf{U}_2, \mathbf{V}_2)) \triangleq \mathbf{U}_1 + \mathbf{U}_2 \mod q$
- $\mathsf{Add}^{out}(\mathbf{V}_1, \mathbf{V}_2) \triangleq \mathbf{V}_1 + \mathbf{V}_2 \mod q$

The addition-noise is bounded by $\beta_1 + \beta_2$. The correctness follows by $(\mathbf{V}_1 + \mathbf{V}_2) = \mathbf{A}(\mathbf{U}_1 + \mathbf{U}_2) + (x_1 + x_2)\mathbf{G}$.

Homomorphic Multiplication Algorithms. The homomorphic *input* multiplication algorithm is asymmetric and involved in whole input, partial output and index, and the homomorphic *output* multiplication algorithm is essentially a multiplicaiton of the output matrices.

- $\mathsf{Multi}^{in}((x_1, \mathbf{U}_1, \mathbf{V}_1), (x_2, \mathbf{U}_2, \mathbf{V}_2)) \triangleq x_2 \cdot \mathbf{U}_1 + \mathbf{U}_2 \cdot \widehat{\mathbf{V}}_1 \mod q$
- $\mathsf{Multi}^{out}(\mathbf{V}_1, \mathbf{V}_2) \triangleq \mathbf{V}_2 \cdot \widehat{\mathbf{V}}_1 \mod q$

The multiplication-noise is bounded by $|x_2|\beta_1 + m\beta_2 = \beta_1 + m\beta_2$. The correctness also follows by a simple computation assuming $\mathbf{V}_i = \mathbf{A}\mathbf{U}_i + x_i\mathbf{G}$.

4.2 The Homomorphic *Output* and *Input* Evaluation

Homomorphic *Output* Evaluation. We define the homomorphic *output* evaluation algorithm

$$\mathsf{Eval}^{out}(\Pi, \mathbf{V}_0, \{\mathbf{V}_{0,i}\}_{i\in[w]}, \{\mathbf{V}_j\}_{j\in[t]}) \to \mathbf{V}_\Pi$$

for a length-L permutation branching program Π, where $\mathbf{V}_0, \{\mathbf{V}_{0,i}\}_{i\in[w]}$ will be assigned in the initialization stage below and $\mathbf{V}_j$ is such that $\mathbf{V}_j = \mathbf{A}\mathbf{U}_j + x_j\mathbf{G}$. Recall that $\{(h(k), \gamma_{k,i,0}, \gamma_{k,i,1})\}_{k\in[L],i\in[w]}$ is a valid description of Π, and that the initial state vector is set to be the first w-dimensional unit vector $\mathbf{v}_0 = (1, 0, 0, \ldots, 0)$, and that for $k \in [L]$ and $i \in [w]$,

$$\mathbf{v}_k[i] = \mathbf{v}_{k-1}[\gamma_{k,i,1}] \cdot x_{h(k)} + \mathbf{v}_{k-1}[\gamma_{k,i,0}] \cdot (1 - x_{h(k)}).$$

The homomorphic *output* evaluation algorithm Eval^{out} proceeds as follows.

- **Initialization:** For $k \in [L], i \in [w]$, let $\mathbf{V}_k[i]$ be an *output* corresponding to the state $\mathbf{v}_k[i]$.
  1. Choose $\mathbf{V}_{0,i} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ uniformly at random and set it be an initial *output* corresponding to the initial state $\mathbf{v}_0[i]$.
  2. Choose $\mathbf{V}_0 \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ uniformly at random and see it be an *output* corresponding to a constant state 1.
  3. Set $\bar{\mathbf{V}}_j \triangleq \mathbf{V}_0 - \mathbf{V}_j$ and see it be an *output* corresponding to $(1 - x_j)$, where $\mathbf{V}_j$ (so that $\mathbf{V}_j = \mathbf{A}\mathbf{U}_j + x_j\mathbf{G}$) is an *output* corresponding to x_j.

– **Computation:** For $k = 1, 2, \ldots, L$, the computation process proceeds inductively as follows. Assume that at step $t-1$, we have $\{\mathbf{V}_{k-1,i}\}_{i\in[w]}$. We compute

$$\mathbf{V}_{k,i} = \mathbf{V}_{h(k)} \cdot \widehat{\mathbf{V}}_{k-1,\gamma_{k,i,1}} + \bar{\mathbf{V}}_{h(k)} \cdot \widehat{\mathbf{V}}_{k-1,\gamma_{k,i,0}}. \tag{5}$$

– **Final *Output***: Finally, we have $\{\mathbf{V}_{L,i}\}_{i\in[w]}$ after finishing the computation process. Output $\mathbf{V}_{L,1}$ as the final *output* corresponding to $\mathbf{v}_L[1]$, i.e., $\mathbf{V}_{\Pi} = \mathbf{V}_{L,1}$.

Homomorphic *Input* Evaluation. We define the homomorphic *input* evaluation algorithm

$$\mathsf{Eval}^{in}(\Pi, (1, \mathbf{U}_0, \mathbf{V}_0), \{(\mathbf{v}_0[i], \mathbf{U}_{0,i}, \mathbf{V}_{0,i})\}_{i\in[w]}, \{(x_j, \mathbf{U}_j, \mathbf{V}_j)\}_{j\in[t]}) \to \mathbf{U}_{\Pi}$$

for a permutation branching program Π which proceeds as follows.

– **Initialization:** For $k \in [L], i \in [w]$, let $\mathbf{U}_k[i]$ be an *input* corresponding to the state $\mathbf{v}_k[i]$.
  1. Sample $\mathbf{U}_{0,i} \leftarrow \mathcal{D}_{\mathcal{U}}$ (such that $\mathbf{V}_{0,i} = \mathbf{A}\mathbf{U}_{0,i} + \mathbf{v}_0[i]\mathbf{G}$) and see it be an initial *input* corresponding to the initial state $\mathbf{v}_0[i]$.
  2. Sample $\mathbf{U}_0 \leftarrow \mathcal{D}_{\mathcal{U}}$ (such that $\mathbf{V}_0 = \mathbf{A}\mathbf{U}_0 + 1 \cdot \mathbf{G}$) and see it be an *input* corresponding to a constant state 1.
  3. Set $\bar{\mathbf{U}}_j \triangleq \mathbf{U}_0 - \mathbf{U}_j$, where $\mathbf{U}_j$ (such that $\mathbf{V}_j = \mathbf{A}\mathbf{U}_j + x_j\mathbf{G}$) is an *input* corresponding to x_j and see it be an *input* corresponding to $(1 - x_j)$.
– **Computation:** For $k = 1, 2, \ldots, L$, the computation process proceeds inductively as follows. Assume that at step $t-1$, we have $\{\mathbf{U}_{k-1,i}\}_{i\in[w]}$. We compute

$$\begin{aligned}\mathbf{U}_{k,i} = (&x_{h(k)} \cdot \mathbf{U}_{k-1,\gamma_{k,i,1}} + \mathbf{U}_{h(k)} \cdot \widehat{\mathbf{V}}_{k-1,\gamma_{k,i,1}}) \\ &+ ((1 - x_{h(k)}) \cdot \mathbf{U}_{k-1,\gamma_{k,i,0}} + \bar{\mathbf{U}}_{h(k)} \cdot \widehat{\mathbf{V}}_{k-1,\gamma_{k,i,0}}).\end{aligned} \tag{6}$$

– **Final *Input***: Finally, we have $\{\mathbf{U}_{L,i}\}_{i\in[w]}$ after finishing the computation process. Output $\mathbf{U}_{L,1}$ as the final *input* corresponding to $\mathbf{v}_L[1]$, i.e., $\mathbf{U}_{\Pi} = \mathbf{U}_{L,1}$.

4.3 Correctness of Homomorphic Evaluation and Noise Analysis

We will prove the correctness of above homomorphic *input-output* evaluation algorithms and analyze the noise growth under homomorphic evaluation.

Lemma 4.1. *Assuming that* $\mathsf{Eval}^{out}(\Pi, \mathbf{V}_0, \{\mathbf{V}_{0,i}\}_{i\in[w]}, \{\mathbf{V}_j\}_{j\in[t]}) \to \mathbf{V}_{\Pi}$ *and* $\mathsf{Eval}^{in}(\Pi, (1, \mathbf{U}_0, \mathbf{V}_0), \{(\mathbf{v}_0[i], \mathbf{U}_{0,i}, \mathbf{V}_{0,i})\}_{i\in[w]}, \{(x_j, \mathbf{U}_j, \mathbf{V}_j)\}_{j\in[t]}) \to \mathbf{U}_{\Pi}$ *are such that* $\mathbf{V}_0 = \mathbf{A}\mathbf{U}_0 + 1 \cdot \mathbf{G}$, $\mathbf{V}_{0,i} = \mathbf{A}\mathbf{U}_{0,i} + \mathbf{v}_0[i]\mathbf{G}$ *and* $\mathbf{V}_j = \mathbf{A}\mathbf{U}_j + x_j\mathbf{G}$ *for* $i \in [w], j \in [t]$. *For all* $k \in [L], i \in [w]$, *we then have*

$$\mathbf{V}_{k,i} = \mathbf{A}\mathbf{U}_{k,i} + \mathbf{v}_k[i]\mathbf{G}.$$

In particular, we have $\mathbf{V}_{L,1} = \mathbf{A}\mathbf{U}_{L,1} + \mathbf{v}_L[1]\mathbf{G}$.

Proof. Given the conditions in this lemma, by formulas (1), (5) and (6), we have

$$\begin{aligned}
\mathbf{A}\mathbf{U}_{k,i} + \mathbf{v}_k[i]\mathbf{G} =& \mathbf{A} \cdot \Big[\big(x_{h(k)} \cdot \mathbf{U}_{k-1,\gamma_{k,i,1}} + \mathbf{U}_{h(k)} \cdot \widehat{\mathbf{V}}_{k-1,\gamma_{k,i,1}}\big) \\
&+ \big((1 - x_{h(k)}) \cdot \mathbf{U}_{k-1,\gamma_{k,i,0}} + \bar{\mathbf{U}}_{h(k)} \cdot \widehat{\mathbf{V}}_{k-1,\gamma_{k,i,0}}\big)\Big] \\
&+ \Big(\mathbf{v}_{k-1}[\gamma_{k,i,1}] \cdot x_{h(k)} + \mathbf{v}_{k-1}[\gamma_{k,i,0}] \cdot (1 - x_{h(k)})\Big) \cdot \mathbf{G} \\
=& \Big(x_{h(k)} \cdot \mathbf{V}_{k-1,\gamma_{k,i,1}} - x_{h(k)} \cdot \mathbf{v}_{k-1}[\gamma_{k,i,1}] \cdot \mathbf{G}\Big) \\
&+ \Big(\mathbf{V}_{h(k)} \cdot \widehat{\mathbf{V}}_{k-1,\gamma_{k,i,1}} - x_{h(k)} \cdot \mathbf{V}_{k-1,\gamma_{k,i,1}}\Big) \\
&+ \Big((1 - x_{h(k)}) \cdot \mathbf{V}_{k-1,\gamma_{k,i,0}} - (1 - x_{h(k)}) \cdot \mathbf{v}_{k-1}[\gamma_{k,i,0}] \cdot \mathbf{G}\Big) \\
&+ \Big(\bar{\mathbf{V}}_{h(k)} \cdot \widehat{\mathbf{V}}_{k-1,\gamma_{k,i,0}} - (1 - x_{h(k)}) \cdot \mathbf{V}_{k-1,\gamma_{k,i,0}}\Big) \\
&+ \Big(x_{h(k)} \cdot \mathbf{v}_{k-1}[\gamma_{k,i,1}] \cdot \mathbf{G} + (1 - x_{h(k)}) \cdot \mathbf{v}_{k-1}[\gamma_{k,i,0}] \cdot \mathbf{G}\Big) \\
=& \mathbf{V}_{h(k)} \cdot \widehat{\mathbf{V}}_{k-1,\gamma_{k,i,1}} + \bar{\mathbf{V}}_{h(k)} \cdot \widehat{\mathbf{V}}_{k-1,\gamma_{k,i,0}} \\
=& \mathbf{V}_{k,i}
\end{aligned}$$

for all $k \in [L], i \in [w]$. This finishes the proof. □

Lemma 4.2. *Assuming that* $\mathsf{Eval}^{in}(\Pi, (1, \mathbf{U}_0, \mathbf{V}_0), \{(\mathbf{v}_0[i], \mathbf{U}_{0,i}, \mathbf{V}_{0,i})\}_{i\in[w]}, \{(x_j, \mathbf{U}_j, \mathbf{V}_j)\}_{j\in[t]}) \to \mathbf{U}_\Pi$ *is such that all the input-noises are bounded by* β, *i.e.,* $\|\mathbf{U}_0\|_\infty, \|\mathbf{U}_{0,i}\|_\infty, \|\mathbf{U}_j\|_\infty \le \beta$, *it then holds that* $\|\mathbf{U}_\Pi\|_\infty \le 3mL\beta + \beta$.

Proof. We will simply show the lemma by inductive method. Namely, we will show that $\|\mathbf{U}_{k,i}\|_\infty \le 3km\beta + \beta$ for any step $k = 0, 1, 2, \ldots, L$ and $i \in [w]$.

If $k = 0$, there is no computation and by initialization it is very easy to see that all the initial noises are such that $\|\mathbf{U}_{0,i}\|_\infty \le \beta, i \in [w]$.

Assume that at step $k-1$, we have $\|\mathbf{U}_{k,i}\|_\infty \le 3m(k-1)\beta + \beta$. By formula (6), we obtain that

$$\begin{aligned}
\|\mathbf{U}_{k,i}\|_\infty =& \|(x_{h(k)} \cdot \mathbf{U}_{k-1,\gamma_{k,i,1}} + \mathbf{U}_{h(k)} \cdot \widehat{\mathbf{V}}_{k-1,\gamma_{k,i,1}}) \\
&+ ((1 - x_{h(k)}) \cdot \mathbf{U}_{k-1,\gamma_{k,i,0}} + \bar{\mathbf{U}}_{h(k)} \cdot \widehat{\mathbf{V}}_{k-1,\gamma_{k,i,0}})\|_\infty \\
\le& \|x_{h(k)} \cdot \mathbf{U}_{k-1,\gamma_{k,i,1}}\|_\infty + \|\mathbf{U}_{h(k)} \cdot \widehat{\mathbf{V}}_{k-1,\gamma_{k,i,1}}\|_\infty \\
&+ \|(1 - x_{h(k)}) \cdot \mathbf{U}_{k-1,\gamma_{k,i,0}}\|_\infty + \|\bar{\mathbf{U}}_{h(k)} \cdot \widehat{\mathbf{V}}_{k-1,\gamma_{k,i,0}}\|_\infty \\
\le& x_{h(k)} \cdot (3m(k-1)\beta + \beta) + m\beta + (1 - x_{h(k)}) \cdot (3m(k-1)\beta + \beta) + 2m\beta \\
=& 3mk\beta + \beta
\end{aligned}$$

where $\|\bar{\mathbf{U}}_{h(k)}\|_\infty = \|\mathbf{U}_0 - \mathbf{U}_{h(k)}\|_\infty \le \|\mathbf{U}_0\|_\infty + \|\mathbf{U}_{h(k)}\|_\infty \le \beta + \beta = 2\beta$.

By induction, we get $\|\mathbf{U}_\Pi\|_\infty = \|\mathbf{U}_{L,1}\|_\infty \le 3mL\beta + \beta$. This finishes the proof. □

Remark. By Barrington's theorem [3], a depth-d circuit can be transformed to a length $L = 4^d$ permutation branching program. Therefore, whenever

$d \leq \text{poly}(\lambda)$, the maximum noise comparing to Gorbunov-Vaikuntanathan-Wichs' HTDF reduces roughly from $O(m^d\beta)$ to $O(4^d m\beta)$. In particular, we can set polynomial modulus $q = \text{poly}(\lambda) > O(4^d m\beta)$ when $d = O(\log \lambda)$ which will result in better security based on GapSVP with polynomial approximation factors.

5 Strongly-Unforgeable Identity-Based Fully Homomorphic Signatures

5.1 Definition

A single data-set identity-based homomorphic signature scheme consists of the following poly-time algorithms ($\mathsf{PrmsGen}$, Setup, $\mathsf{Extract}$, Sign, $\mathsf{SignEval}$, $\mathsf{Process}$, Verify) with syntax:

- $prms \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N)$: Take the security parameter λ and the maximum data-size N. Output public parameters $prms$. The security parameter also defines the message space $\mathcal{X}$.
- $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$: Take the security parameter λ. Output a master key pair (mpk, msk).
- $(pk_{id}, sk_{id}) \leftarrow \mathsf{Extract}(mpk, msk, id)$: An identity-key extraction procedure.
- $(\sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk_{id}}(prms, x_1, \ldots, x_N)$: Sign message data $(x_1, \ldots, x_N) \in \mathcal{X}^N$ to id.
- $\sigma_g = \mathsf{SignEval}_{prms}(g, (x_1, \sigma_1), \ldots, (x_t, \sigma_t))$: Deterministically and homomorphically evaluate a signature σ_g for some function g over $(x_1, \ldots, x_t) \in \mathcal{X}^t$.
- $v_g = \mathsf{Process}_{prms}(g)$: Deterministically and homomorphically evaluate a *certificate* v_g for the function g from the public parameters $prms$.
- $\mathsf{Verify}_{pk_{id}}(v_g, y, \sigma_g)$: Verify that y is the correct output of g by proving σ_g corresponding to v_g.

Correctness. For $prms \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N)$, $(pk_{id}, sk_{id}) \leftarrow \mathsf{Extract}(mpk, msk, id)$, $(x_1, \ldots, x_N) \in \mathcal{X}^N$, $(\sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk_{id}}(prms, x_1, \ldots, x_N)$, and $g : \mathcal{X}^N \to \mathcal{X}$, we require that the following equation

$$\mathsf{Verify}_{pk_{id}}(v_g, y = g(x_1, \ldots, x_N), \sigma_g) = \text{accept}$$

holds, where $v_g = \mathsf{Process}_{prms}(g)$ and $\sigma_g = \mathsf{SignEval}_{prms}(g, (x_1, \sigma_1), \ldots, (x_t, \sigma_t))$.

Relaxation Correctness of Leveled IBFHS. Here, the relaxation correctness of leveled IBFHS follows from that of leveled IBHTDF and hence is omitted.

Security Experiment. The experiment $\mathbf{Exp}^{\text{SU-sID-sCMA}}_{\mathcal{A},\mathsf{IBFHS}}(1^\lambda)$ defined in Fig. 2 describes the *strongly-unforgeable selective-identity static chosen-message-attack* security game, where the adversary has to fix a target identity id^* to attack and message data to sign before obtaining the master public-key and public parameters. Moreover, the adversary can query identity-keys for all identities

except id^*. He is then forced to find (g, y', σ') such that the winning conditions (described in the experiment) hold. Remark that we do not require either $y = y'$ or not. So, if $y = y'$, then σ' is a strongly-forgeable signature, otherwise a existentially-forgeable signature.

$$\mathbf{Exp}^{\text{SU-sID-sCMA}}_{\mathcal{A},\mathsf{IBFHS}}(1^\lambda)$$

- $(id^*, \{x_i\}_{i\in[N]}, state) \leftarrow \mathcal{A}(1^\lambda)$
- $prms \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N), (mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$
- $(g, y', \sigma') \leftarrow \mathcal{A}^{\mathsf{Extract}(mpk,msk,\cdot)\setminus\{id^*\},\mathsf{Sign}(id^*,\{x_i\}_{i\in[N]})}(prms, mpk, state)$
- $\mathcal{A}$ wins if all of the following hold:
  1. g is admissible on the messages $x_1, \ldots, x_N$;
  2. $\sigma' \neq \sigma_g$, where $\sigma_g = \mathsf{SignEval}_{prms}(g, (x_1, \sigma_1), \ldots, (x_N, \sigma_N))$;
  3. $\mathsf{Verify}_{pk_{id^*}}(v_g, y', \sigma')$ accept, where $v_g = \mathsf{Process}_{prms}(g)$.

Fig. 2. Definition of security for IBFHS with single data-set

We say an IBFHS is *strongly-unforgeable selective-identity static chosen-message-attack* (SU-sID-sCMA) secure if $\Pr[\mathbf{Exp}^{\text{SU-sID-sCMA}}_{\mathcal{A},\mathsf{IBFHS}}(1^\lambda)] \leq \mathrm{negl}(\lambda)$.

5.2 Construction

Let $\mathcal{F} = (\mathsf{IBHTDF.Setup}, \mathsf{IBHTDF.Extract}, f, \mathsf{Invert}, \mathsf{IBHTDF.Eval}^{in}, \mathsf{IBHTDF.Eval}^{out})$ be an IBHTDF with identity space $\mathcal{I}$, index space $\mathcal{X}$, input space $\mathcal{U}$, output space $\mathcal{V}$ and some efficiently samplable input distribution $\mathcal{D}_\mathcal{U}$ over $\mathcal{U}$. We construct an IBFHS scheme $\mathcal{S} = (\mathsf{PrmsGen}, \mathsf{Setup}, \mathsf{Extract}, \mathsf{Sign}, \mathsf{SignEval}, \mathsf{Process}, \mathsf{Verify})$ with message space $\mathcal{X}$ as follows.

- $prms \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N)$: Sample $v_i \xleftarrow{\$} \mathcal{V}, i \in [N]$ and set public parameters $prms = (v_1, \ldots, v_N)$.
- $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$: Select $(mpk', msk') \leftarrow \mathsf{IBHTDF.Setup}(1^\lambda)$ and set master-key pair $(mpk = mpk', msk = msk')$.
- $(pk_{id}, sk_{id}) \leftarrow \mathsf{Extract}(mpk, msk, id)$: Run $\mathsf{IBHTDF.Extract}(mpk', msk', id)$ to get (pk'_{id}, sk'_{id}) and set $pk_{id} = pk'_{id}, sk_{id} = sk'_{id}$ for $id \in \mathcal{I}$.
- $(\sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}_{sk_{id}}(prms, x_1, \ldots, x_N)$: Sample $u_i \leftarrow \mathsf{Invert}_{sk'_{id},x_i}(v_i)$ and set $\sigma_i = u_i, i \in [N]$.
- $\sigma_g = \mathsf{SignEval}_{prms}(g, (x_1, \sigma_1), \ldots, (x_t, \sigma_t))$: Perform deterministic algorithm $\mathsf{IBHTDF.Eval}^{in}(g, (x_1, u_1, v_1), \ldots, (x_t, u_t, v_t))$ to get u_g and set $\sigma_g = u_g$.
- $v_g = \mathsf{Process}_{prms}(g)$: Perform $\mathsf{IBHTDF.Eval}^{out}(g, v_1, \ldots, v_t)$ and output the result v_g.
- $\mathsf{Verify}_{pk_{id}}(v_g, y, \sigma_g)$: If $f_{pk'_{id},y}(\sigma_g) = v_g$ accept, else reject.

Correctness. Here, the discussion of the relaxation correctness of the leveled IBFHS constructed above follows from that of the underlying leveled IBHTDF in Sect. 3 and hence is omitted.

Security. We now show the SU-sID-sCMA security of the leveled IBFHS above.

Theorem 5.1. *The leveled* IBFHS *scheme* $\mathcal{S}$ *constructed above is* SU-sID-sCMA *secure assuming that* $\mathcal{F}$ *is a leveled selective-identity secure* IBHTDF.

Proof. Assume there exists a PPT adversary $\mathcal{A}$ that wins the security experiment $\mathbf{Exp}^{\text{SU-sID-sCMA}}_{\mathcal{A},\mathsf{IBFHS}}(1^\lambda)$ of IBFHS with non-negligible probability δ. We construct a PPT reduction $\mathcal{B}$ that breaks the selective-identity security of $\mathcal{F}$.

Let id^* be the identity that $\mathcal{A}$ intends to attack. $\mathcal{B}$ will run the changed algorithms ($\mathsf{PrmsGen}^*$, Setup^*, $\mathsf{Extract}^*$, Sign^*).

- $\mathsf{Setup}^*(1^\lambda)$: Run $(mpk', msk') \leftarrow \mathsf{IBHTDF.Setup}^*(1^\lambda)$ and set $mpk = mpk'$, $msk = msk'$.
- $\mathsf{Extract}^*(mpk, msk, id)$: Run $(pk'_{id}, sk'_{id}) \leftarrow \mathsf{IBHTDF.Extract}^*(mpk, \mathbf{R}, id)$ when $id \neq id^*$ and set $pk_{id} = pk'_{id}, sk_{id} = sk'_{id}$. However, if $id = id^*$, then the trapdoor disappears and $\mathcal{B}$ can not generate identity key for id^*.
- $\mathsf{PrmsGen}^*(1^\lambda, 1^N)$: Choose $u_i \leftarrow \mathcal{D}_\mathcal{U}$ and compute $v_i = f_{pk_{id^*}, x_i}(u_i)$. Output $prms = (v_1, \ldots, v_N)$.
- $\mathsf{Sign}^*(x_1, \ldots, x_N)$: Set $\sigma_i = u_i$ and output $(\sigma_1, \ldots, \sigma_N)$.

The views of adversary $\mathcal{A}$ between the original experiment and the changed experiment are indistinguishable by *Distributional Equivalence of Inversion* property of the underlying IBHTDF. In particular, the winning probability of $\mathcal{A}$ attacking the changed experiment is at least $\delta - \text{negl}(\lambda)$.

We now show that there exists a PPT reduction $\mathcal{B}$ that takes any PPT adversary $\mathcal{A}$ winning the changed experiment with non-negligible advantage $\delta - \text{negl}(\lambda)$, and that breaks the $\mathbf{Exp}^{\text{sID}}_{\mathcal{A},\mathsf{IBHTDF}}(1^\lambda)$ security of the underlying $\mathcal{F}$ with probability $\delta - \text{negl}(\lambda)$.

The reduction $\mathcal{B}$ receives the challenge identity id^* and message data-set $(x_1, \ldots, x_N)$, generates $(mpk, msk, \{\sigma_i = u_i, v_i\}_{i\in[N]})$ as in the changed experiment and sends $(mpk, \{\sigma_i, v_i\}_{i\in[N]})$ to $\mathcal{A}$. Note that $\mathcal{B}$ can respond to the identity-key query for $id \neq id^*$ using msk. But, $\mathcal{B}$ has no valid trapdoor to generate the identity key for id^*.

Assume the adversary $\mathcal{A}$ (winning the changed experiment) outputs values (g, y', σ'), where $g : \mathcal{X}^N \to \mathcal{X}$ on $(x_1, \ldots, x_N)$ is an admissible function and $\sigma' = u'$. Let $y = g(x_1, \ldots, x_N), u_g = \sigma_g = \mathsf{SignEval}_{prms}(g, (x_1, \sigma_1), \ldots, (x_t, \sigma_t))$, $v_g = \mathsf{Process}_{prms}(g)$. Thus, on one hand, since the forged signature σ' verifies, $f_{pk_{id^*}, y'}(u') = v_g$ holds. On the other hand, since g is admissible, $f_{pk_{id^*}, y}(u_g) = v_g$ also holds by the correctness of homomorphic computation. Therefore, we have values $u_g \neq u' \in \mathcal{U}$ and $y, y' \in \mathcal{X}$ satisfying $f_{pk_{id^*}, y}(u_g) = f_{pk_{id^*}, y'}(u')$, which allows $\mathcal{B}$ to break $\mathbf{Exp}^{\text{sID}}_{\mathcal{A},\mathsf{IBHTDF}}(1^\lambda)$ security of $\mathcal{F}$ with probability $\delta - \text{negl}(\lambda)$ whenever $\mathcal{A}$ wins the changed experiment with probability $\delta - \text{negl}(\lambda)$. □

6 Conclusions

In this work, we defined and constructed the first leveled strongly-unforgeable IBFHS schemes. To this end, we extended Gorbunov-Vaikuntanathan-Wichs'

HTDF, the underlying primitive of FHS, to IBHTDF with stronger security and better parameters, the underlying primitive of IBFHS. The drawback is that our scheme is only a leveled IBFHS with large public parameters. It remains open to Construct a non-leveled IBFHS or a leveled IBFHS with short public parameters. One way to achieve this would be to draw on the ideas in constructing non-leveled (IB)FHEs from indistinguishability obfuscation [13,14].

Acknowledgement. We are very grateful to the anonymous ISC reviewers for valuable comments and constructive suggestions that helped to improve the presentation of this work.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010)
2. Attrapadung, N., Libert, B.: Homomorphic network coding signatures in the standard model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 17–34. Springer, Heidelberg (2011)
3. Barrington, D.A.M.: Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In: STOC 1986, pp. 1–5. ACM (1986)
4. Boneh, D., Freeman, D.M.: Homomorphic signatures for polynomial functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (2011)
5. Boneh, D., Freeman, D.M.: Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 1–16. Springer, Heidelberg (2011)
6. Boyen, X., Fan, X., Shi, E.: Adaptively secure fully gomomorphic signatures based on lattices. http://eprint.iacr.org/2014/916
7. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014)
8. Boyen, X.: Lattice mixing and vanishing trapdoors: a framework for fully secure short signatures and more. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 499–517. Springer, Heidelberg (2010)
9. Brakerski Z., Vaikuntanathan, V.: Lattice-based FHE as secure as PKE. In: ITCS, pp. 1–12 (2014)
10. Catalano, D., Fiore, D., Warinschi, B.: Efficient network coding signatures in the standard model. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 680–696. Springer, Heidelberg (2012)
11. Catalano, D., Fiore, D., Warinschi, B.: Homomorphic signatures with efficient verification for polynomial functions. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 371–389. Springer, Heidelberg (2014)
12. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)

13. Canetti, R., Lin, H., Tessaro, S., Vaikuntanathan, V.: Obfuscation of probabilistic circuits and applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 468–497. Springer, Heidelberg (2015)
14. Clear, M., McGoldrick, C.: Bootstrappable identity-based fully homomorphic encryption. In: Gritzalis, D., Kiayias, A., Askoxylakis, I. (eds.) CANS 2014. LNCS, vol. 8813, pp. 1–19. Springer, Heidelberg (2014)
15. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. SIAM J. Comput. **38**(1), 97–139 (2008)
16. Freeman, D.M.: Improved security for linearly homomorphic signatures: a generic framework. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 697–714. Springer, Heidelberg (2012)
17. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC 2009, pp. 169–178. ACM (2009)
18. Gorbunov, S., Vaikuntanathan, V., Wichs, D.: Leveled fully homomorphic signatures from standard lattices. In: STOC 2015, pp. 469–477. ACM (2015)
19. Micciancio, D., Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012)
20. Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 21–39. Springer, Heidelberg (2013)
21. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM J. Comput. **37**(1), 267–302 (2007)
22. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
23. Xie, X., Xue, R.: Bounded fully homomorphic signature schemes. http://eprint.iacr.org/2014/420