Hindawi
Security and Communication Networks
Volume 2022, Article ID 9942717, 11 pages
https://doi.org/10.1155/2022/9942717

Research Article

NTRU-Based Fully Homomorphic Signature

Ruonan Li,[1,2] Fuqun Wang ,[1,2,3] Renjun Zhang,[1,2] and Kefei Chen[1,2,3]

[1]*School of Mathematics, Hangzhou Normal University, Hangzhou 311121, China*
[2]*Key Laboratory of Cryptography of Zhejiang Province, Hangzhou Normal University, Hangzhou 311121, China*
[3]*Westone Cryptologic Research Center, Beijing 100071, China*

Correspondence should be addressed to Fuqun Wang; fqwang@hznu.edu.cn

Received 28 November 2021; Accepted 13 May 2022; Published 22 June 2022

Academic Editor: Iqtadar Hussain

Although the fully homomorphic signature (FHS) has made great progress, the low efficiency of existing FHS schemes as one main drawback cannot reach the requirements in practical application. It is well known that Number-Theory-Research-Unit (NTRU)-based cryptography is not only considered to be resistant to the quantum computer but also has high efficiency compared with the standard lattice-based cryptographic systems. In this paper, our goal is to construct a simple and highly efficient FHS scheme. To this end, we first devise an NTRU-based homomorphic trapdoor function (HTDF), which is more effective than the one proposed by Gorbunov et al. (STOC 2015). We then convert it into an efficient NTRU-based FHS scheme. Additionally, the HTDF we constructed is not only collision-resistant but is also claw-free, resulting in our FHS scheme is being strongly unforgeable.

1. Introduction

As the advantages of cloud computing increase, fully homomorphic cryptography has aroused great interest due to its remarkable characteristics. It not only allows the client to safely upload his/her data to some unreliable clouds but also allows the server to perform calculations on the data. Fully homomorphic cryptography consists of fully homomorphic encryption (FHE) and fully homomorphic signature (FHS). Gentry [1] designed the first fully homomorphic encryption system at STOC 2009 and Gorbunov et al. [2] constructed the first leveled fully homomorphic signature scheme at STOC 2015. In this work, we mainly discuss the issues related to FHS.

Rivest [3] first proposed the concept of homomorphic signature (HS) in 2000. Two years later, Johnson et al. [4] gave a security model and designed several provable secure HS schemes. In a HS scheme, a user signs a data $\mathbf{m} = (m_1, \ldots, m_k)$ in the message space $\mathscr{M}$ using his/her signing key and then distributes the signed data $\sigma = (\sigma_1, \ldots, \sigma_k)$ to some unreliable remote server. The server can run arbitrary computations $y = g(\mathbf{m})$ over the signed data σ and homomorphically derive a short signature $\sigma_{g,y}$, which can verify that y is indeed the correct output of g operation on message $\mathbf{m}$. Anyone can check the tuple $(g, y, \sigma_{g,y})$ by employing the user's public verification key and then determine that y is indeed the correct outcome of $g' s$ calculation on $\mathbf{m}$. In addition, the verification process does not need the underlying data $\mathbf{m}$.

According to the homomorphic capability of HS schemes, we can divide the homomorphic signature schemes into three categories: linear homomorphic signature, somewhat homomorphic signature (SHS), and fully homomorphic signature.

Zhao et al. [5] proposed a linear HS scheme to resolve the inherent network pollution problem in network coding. This scheme allows an arbitrary linear combination calculation of the signature data, which can both conveniently verify the integrity of the received message and effectively prevent the application built on the network code from being attacked by pollution. In further research studies, many papers [6–11] have made further improvements in efficiency, security, and privacy protection.

In 2011, Boneh and Freeman [12] constructed the first SHS scheme, which is selectively secure in the random oracle (RO) model. Subsequently, Catalano et al. [13] proposed a new SHS scheme, which is adaptively secure, whose security no longer depends on RO model, and whose verification process is more efficient.

Different from the abovementioned homomorphic signature schemes, the FHS allows polynomial depth circuit operations on the signed data. In 2014, Gorbunov et al. [2] proposed the first leveled FHS scheme based on the hardness of the small integer solution (SIS) problem in standard lattices [14, 15]. To this end, a building block called the homomorphic trapdoor function (HTDF) is proposed. This new cryptographic primitive helps us to conceptually unify digital signatures [16–18] and fully homomorphic encryption [19]. They showed that if an HTDF function is claw-free, then you can directly use the HTDF function to construct an existentially unforgeable FHS scheme under a selective chosen-message attack (EU-sCMA), and that an EU-sCMA secure FHS can be converted into an existentially unforgeable under the adaptive chosen-message attack (EU-aCMA) one with the help of HTDF functions. Additionally, this solution has the following advantages: First, it allows quick amortization verification of calculations on multiple different datasets, even though these datasets belong to different users with different verification keys. Second, the scheme can be made context hiding, that is, the signature $\sigma_{g,y}$ corresponding to $y = g(\mathbf{m})$ will not expose any information about the underlying data $\mathbf{m}$. Third, the scheme also allows several different calculations to be combined on signed data. Inspired by the key-homomorphic function encryption of the circuit, Boyen et al. [20] constructed the first adaptively secure homomorphic signature scheme utilizing the vanishing trapdoor technique [21]. Compared with Gorbunov et al. FHS, the efficiency of this scheme has been significantly improved, and it can fight against stronger adversaries, but this scheme does not reach context hiding. Later, Tsabary [22] demonstrated the equivalence between FHS and attribute-based signature (ABS) [23]. They can be implied by each other under certain conditions, which opens up a new direction for constructing FHS schemes. Furthermore, Tsabary built an ABS from lattices and converted it into a new FHS scheme. All the abovementioned existing HS algorithms can only produce one signature for a single message at a time, which is inefficient. Therefore, Luo et al. [24] proposed a new leveled fully homomorphic signature scheme, which generates signatures for all messages in the dataset only by calling the signature algorithm twice. To the end, the vector coding technique [25] is used, which encodes all messages in the dataset into two matrices.

In addition, Wang et al. [26] constructed an identity-based HTDF (IBHTDF), which has better parameters and stronger security. The maximum noise compared with Gorbunov et al. HTDF is roughly reduced from $O(m^d\beta)$ to $O(4^d m\beta)$, which will lead to a polynomial modulus $q = \text{ploy}(\lambda)$ when $d = O(\log\lambda)$, where d is the maximum depth of the circuit and λ is the security parameter. The stronger security requires that IBHTDF be not only claw-free but also collision-resistant. Then, they converted the abovementioned IBHTDF into a leveled strongly unforgeable IBFHS scheme. However, the leveled IBFHS scheme is only secure in the selective security model under the hardness of the SIS problem. Recently, Wang et al. [27] used trapdoor vanishing [21] and vector coding techniques [25] to construct an efficient leveled strongly unforgeable IBFHS scheme based on the SIS problem on arbitrary lattices. Compared with [26], this scheme is adaptively secure against chosen identity and chosen message attacks under the standard SIS assumption. Concurrently, Wang and Wang [28] constructed a new leveled strongly unforgeable IBFHS scheme based on a new IBHTDF, whose homomorphic multiplication operation is very similar to the homomorphic addition operation, resulting in that the noise level of IBHTDF is used to evaluate circuits with depth d was reduced from $O(4^d m\beta)$ in [26] to $O(2^d\beta)$.

1.1. Motivation and Contribution.

Fully homomorphic signature is an important branch of digital signatures, which plays a significant role in the information age. At present, many achievements have been made in this research field. To our best knowledge, the existing FHS schemes are constructed on the standard lattices. They are not practical because of their low efficiency. Therefore, it is intriguing to design more efficient FHS schemes.

In 1998, Hoffstein et al. [29] put forward a new public key cryptography system called NTRU. NTRU features relatively short, easy to create key, high speed, and low memory requirements. The security of the NTRU cryptosystem comes from the interaction of the polynomial hybrid system and the independence of the reduced modulus of two relatively prime integers p and q. Stehlé and Steinfeld [30] present a provably secure NTRU-based signature (denoted by NTRUSign in this work) scheme, which is a novel adaption of the unfledged and heuristic NTRU-based signature scheme [31] and the standard lattice-based signature scheme [16]. However, NTRUSign has no homomorphic property. Therefore, it is necessary to design an NTRU-based FHS.

Following the blueprint of [2, 26], we first devise an NTRU-based HTDF function from the NTRUSign proposed by Stehlé et al., which is both collision-resistant and claw-free. We then convert it into a strongly unforgeable FHS scheme.

We point out that when only performing polylog-depth circuits, we can compute them by Barrington's theorem [32]. Hence, a polynomial modulus is enough, resulting in improved efficiency and security [26]. In addition, we also could choose the ring $[X]/(X^{n-1} + \cdots + X + 1)$ used in [33] to construct an FHS scheme with similar efficiency.

1.2. Paper Organization.

In Section 2, we provide some background used in this work. We then formally describe our NTRU-based HTDF functions in Section 3 and NTRU-based FHS in Section 4. Finally, we conclude in Section 5.

2. Preliminaries

Throughout, we use the bold lowercase letters (e.g., $\overrightarrow{a}$) to denote row vectors. We use the $\overrightarrow{a}[i]$ to denote the i-th entry of $\overrightarrow{a}$ and $\|\overrightarrow{a}\|_\infty = \max_i \|\overrightarrow{a}[i]\|_\infty$ to represent the infinity norm of $\vec{a}$. Sometimes, we denote $\vec{u} = (\vec{u}_1, \vec{u}_2) \in R^{2\times\ell}$ and $\vec{u} = [\vec{u}_1, \vec{u}_2] \in R^{2\ell}$. We let $[n]$ denote the set $[n] = \{1, 2, \ldots, n\}$.

Let $R[x]/\langle\phi(x)\rangle$ denote the rings for some integer polynomials $\phi(x) \in [x]$ of degree n. In this paper, we use the cyclotomic polynomial $\phi(x) = x^n + 1$, where $n = 2^k$ for positive integer k and $R_q \triangleq R/qR$ for some integer q which is isomorphic to $\mathbb{Z}_q[x]/\langle\phi(x)\rangle$. We represent elements in R_q as $a(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$, where its coefficients are in the range $(-q/2, q/2]$ and the infinity norm of a is $\|a\|_\infty = \max|a_i|$.

Then, we introduced a parameter λ and a negligible function negl (λ). λ is used to denote the security parameter. The growth negl (λ) is slower than λ^{-c} for any constant $c > 0$ and any sufficiently large value of λ. An event with an overwhelming probability is equivalent to its occurrence with a probability of at least $1 - \text{negl}(\lambda)$.

2.1. Basic Notions

2.1.1. Distributions. For a probability distribution $\mathcal{D}$, we use $x \leftarrow \mathcal{D}$ to denote that x is sampled according to $\mathcal{D}$. For a set S, we use $y \overset{\$}{\leftarrow} S$ to denote that y is sampled uniformly at random from S. For two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, $\mathcal{D}_1 \approx_c \mathcal{D}_2$ and $\mathcal{D}_1 \approx_s \mathcal{D}_2$ denote that $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable and statistically indistinguishable, respectively.

2.1.2. Entropy. 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 follows:

$$\begin{aligned}\widetilde{\mathbf{H}}_\infty(X|Y) &\triangleq -\log\left(\mathbf{E}_{y\leftarrow\mathcal{Y}}\left[\max_x \mathbf{Pr}[X = x|Y = y]\right]\right) \\ &= -\log\left(\mathbf{E}_{y\leftarrow\mathcal{Y}}\left[2^{-\mathbf{H}_\infty(X|Y=y)}\right]\right).\end{aligned} \tag{1}$$

Given the correlation value Y, the optimal probability of an infinite attacker predicting X is $2^{-\widetilde{\mathbf{H}}_\infty(X|Y)}$.

2.1.3. B-Bounded. A distribution ensemble $\{\chi_k\}_{k\in\mathbb{N}}$, supported over the ring R, is called B-bounded if $\Pr_{a\leftarrow\chi_k}[\|a\|_\infty \leq B] = 1$.

Lemma 1 (see [34]). *In a ring $R = \triangleq [x]/\langle x^n + 1\rangle$, for any two polynomials $a, b \in R$, we have the following norm $\|ab\|_\infty \leq n\|a\|_\infty \cdot \|b\|_\infty$.*

Lemma 2 (see [34]). *In a ring $R = \triangleq [x]/\langle x^n + 1\rangle$, let χ be a B-bounded distribution over R, let $a_1, \ldots, a_m \leftarrow \chi$. Then, the value of $\prod_{i=1}^m a_i$ is $n^{m-1}B^m$ bounded.*

2.2. Gaussian and Discrete Gaussian. For $r > 0$, the Gaussian function $\rho_{r,\vec{c}}(\vec{x})$, supported on $\mathbb{R}^n$ and centered at $\vec{c}$ with parameter r and $x \in \mathbb{R}^n$, is defined as follows: $\rho_{r,\vec{c}}(\vec{x}) \triangleq e^{-\pi\|\vec{x}-\vec{c}\|^2/r^2}$.

2.2.1. Discrete Gaussian Distribution. For $r > 0$, the discrete Gaussian function $\rho_{r,\vec{c}}(\vec{x})$, supported on $\mathbb{Z}^n$ and centered at $\vec{c}$ with parameter r and $x \in \mathbb{Z}^n$, is defined as follows: $D_{\mathbb{Z}^n,r,\vec{c}} \triangleq \rho_{r,\vec{c}}(\vec{x})/\rho_{r,\vec{c}}(\mathbb{Z}^n)$. When c equals 0, it is denoted by $D_{\mathbb{Z}^n,r}$. Take note that the following Lemma illustrates that a discrete Gaussian distribution $D_{\mathbb{Z}^n,r}$ outputs a $r\sqrt{n}$-bounded polynomial with overwhelming probability. So, we define a $r\sqrt{n}$-bounded distribution called truncated Gaussian distribution which is statistically indistinguishable from discrete Gaussian distribution.

Lemma 3 (see [34]). *For $n = \omega(\log\lambda)$ and $r > \omega(\sqrt{\log n})$, it holds that $Pr_{x\leftarrow D_{\mathbb{Z}^n,r}}[\|x\| > r\sqrt{n}] \leq 2^{-n+1} = negl(\lambda)$.*

The truncated discrete Gaussian distribution centered at 0 and denoted by $\overline{D}_{\mathbb{Z},r}$ is that sample polynomials from $D_{\mathbb{Z}^n,r}$, and if the polynomial is not $r\sqrt{n}$-bounded, then discards it and repeats the sampling. According to Lemma 3, if $n = \omega(\log\lambda)$ and $r > \omega(\sqrt{\log n})$, then we have $\overline{D}_{\mathbb{Z},r} \approx_s D_{\mathbb{Z}^n,r}$.

2.3. Lattices

2.3.1. RSIS. Choose m polynomials $\vec{a} = (\vec{a}_1, \ldots, \vec{a}_m) \in R_q^m$ uniformly and independently, find $\vec{x} \in R_q^m/0$ such that $\sum_i \vec{a}_i\vec{x}_i = 0 \bmod q$ and $\|\vec{x}\| \leq \beta$, then this problem is called RSIS q, m, β.

The following theorem shows that there is a reduction from the average-case hardness of RSIS to the worst-case hardness of ideal-SVP as follows:

Theorem 1 (see [30]). *For $n = 2^k$, $\phi(x) = x^n + 1$ and $\varepsilon > 0$. Let $m, q > 0$ such that $q > \beta\sqrt{n}\cdot\omega(\log n)$ and $m, \log q \leq poly(n)$. A polynomial-time algorithm solving $RSIS_{q,m,\beta}$ with overwhelming probability can be used to solve Ideal-SVP$_\gamma$ in polynomial time with the approximate factor $\gamma \geq \beta\sqrt{n}\cdot\omega(\sqrt{\log n})$.*

2.3.2. Lattice Trapdoor. In this paper, for an odd integer q and $\ell = (\log q)$, we use the "powers of two" gadget $\vec{g} = (1, 2, \ldots, 2^{\ell-1}) \in R_q^\ell$ first introduced by Micciancio and Peikert. [18]. For $\vec{a} \in R_q^\ell$, $\vec{g}^{-1}(\cdot)$ is the deterministic bit decomposition function, which outputs a matrix $\mathbf{X} \in R_2^{\ell\times\ell}$ so as to $\mathbf{X} = \vec{g}^{-1}(\vec{a})$ and $\vec{g}\cdot\mathbf{X} = \vec{a}$.

Lemma 4 (see [30]). *іere are three efficient algorithms as follows:*

(1) *TrapGen$(1^n, q, \sigma)$: It runs the NTRUSign key generation algorithm same as [30] and returns (h, td).*

Recall that $h = gf^{-1} \in R_q^*$ is a signing key and $td = \begin{pmatrix} f & g \\ F & G \end{pmatrix}$ is the corresponding trapdoor R-basis for the R module $h^\perp = \{u = (u_1, u_2) \in R^{1\times 2}: [h, -1]u = hu_1 - u_2 = 0 \bmod q\}$.

(2) *SamDom*$(1^{2\ell}, q)$*: It samples* $\vec{u} = (\vec{u}_1, \vec{u}_2) \in R^{2\times\ell}$ *such that* $\|\vec{u}\|_\infty \leq \beta_{sam}$ *with probability 1.*

(3) *SamPre*$(h, \vec{v}, td)$*: Given* $h \in R_q^*$, $\vec{v} \in R_q^\ell$ *and the trapdoor td, it returns* $\vec{u} = (\vec{u}_1, \vec{u}_2) \in R^{2\times\ell}$ *such that* $[h, -1]\vec{u} = h\vec{u}_1 - \vec{u}_2 = \vec{v} \bmod q$ *and* $\|\vec{u}\|_\infty \leq \beta_{sam}$ *with probability 1.*

(4) *We have the statistical indistinguishability:*

$$h \approx_s h' \text{and} (h, td, \vec{u}, \vec{v}) \approx_s (h, td, \vec{u}', \vec{v}'), \tag{2}$$

where $(h,td) \leftarrow TrapGen(1^n, q, \sigma), h' \overset{\$}{\leftarrow} R_q^*$, *and* $\vec{u} \leftarrow$ *SamDom* $(1^{2\ell}, q), \vec{v} = [h, -1]\vec{u} \bmod q, \vec{v}' \overset{\$}{\leftarrow} R_q^\ell, \vec{u}' \leftarrow$ *SamPre*$(h, \vec{v}', td)$.

2.4. Permutation Branching Program. Then, a permutation branching program Π is defined as follows [35]. Π with input space of $\{0,1\}^t$, length L, and width w is a sequence of L tuples of the form $(r(k), \sigma_{k,0}, \sigma_{k,1})$ where.

r: $[L] \longrightarrow [t]$ is a function associates the k-th tuple with an input bit $x_{r(k)}$.

$\sigma_{k,0}, \sigma_{k,1}$ are permutations over $[w] = \{1, 2, \ldots, w\}$.

Π performs calculation 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 $\eta_k \in [w]$. Then, the state η_k is calculated recursively according to the formula given below.

$$\eta_k = \sigma_{k, x_{r(k)}}(\eta_{k-1}). \tag{3}$$

After L steps, the final state is η_L. If $\eta_L = 1$, the output of Π is 1, otherwise it is 0.

In order to handle the problem of excessive noise growth caused by homomorphic operation, we present the state in the form of bits, as described in [35]. Specifically, for $\eta_0 = 1$, we use an w-dimensional unit vector $\vec{e}_k$, e.g., $\vec{e}_0 = (1, 0, 0, \ldots, 0)$ instead of the state $\eta_k \in [w]$. The principal idea of this design is that $\vec{e}_k[i] = 1$ holds only when $\sigma_{k, x_{r(k)}}(\eta_{k-1}) = i$. If $\vec{e}_k[i] = 1$ holds, if and only if either:

$x_{r(k)} = 1$ and $\vec{e}_{k-1}[\sigma_{k,1}^{-1}(i)] = 1$ or

$x_{r(k)} = 0$ and $\vec{e}_{k-1}[\sigma_{k,0}^{-1}(i)] = 1$.

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

$$\begin{aligned}\vec{e}_k[i] &= \vec{e}_{k-1}\left[\sigma_{k,1}^{-1}(i)\right] \cdot x_{r(k)} + \vec{e}_{k-1}\left[\sigma_{k,0}^{-1}(i)\right] \cdot \left(1 - x_{r(k)}\right) \\ &= \vec{e}_{k-1}\left[\xi_{k,i,1}\right] \cdot x_{r(k)} + \vec{e}_{k-1}\left[\xi_{k,i,0}\right] \cdot \left(1 - x_{r(k)}\right),\end{aligned} \tag{4}$$

where $\xi_{k,i,1} \triangleq \sigma_{k,1}^{-1}(i)$ and $\xi_{k,i,0} \triangleq \sigma_{k,0}^{-1}(i)$ are fully determined by the description of Π and can be computed easily and publicly. In order to make the homomorphic operation more concise and efficient, we give another representation of permutation branching program, such that $\{(r(k), \xi_{k,i,0}, \xi_{k,i,1})\}_{k\in[L], i\in[w]}$.

3. HTDF Functions

A homomorphic trapdoor function (HTDF) allows anyone to calculate a homomorphic input u_g and an output v_g. The input depends only on $(g, (x_1, u_1, v_1), \ldots, (x_t, u_t, v_t))$ and the output depends on $(g, v_1, \ldots, v_t)$. That is, $f_{pk, g(x_1, \ldots, x_t)}(u_g) = v_g$ holds, where g is an admissible function (or called circuit) defined very soon.

3.1. Definition. A HTDF consists of five polytime algorithms (HTDF.KeyGen, f, Inv, HTDF.Eval$^{\text{in}}$, HTDF.Eval$^{\text{out}}$) with syntax as follows:

$(pk, sk) \leftarrow$HTDF.KeyGen(1^λ): A key generation procedure. The security parameter λ defines the input space $\mathcal{U}$, the index space $\mathcal{X}$, and the output space $\mathcal{V}$ and some efficiently sampleable input-distribution $\mathcal{D}_\mathcal{U}$ over $\mathcal{U}$. We require that elements in $\mathcal{U}, \mathcal{V}$, and $\mathcal{X}$ can be efficiently tested and that elements in $\mathcal{V}$ can efficiently be sampled uniformly at random.

$f_{pk,x}$: $\mathcal{U} \longrightarrow \mathcal{V}$: A deterministic function indexed by pk and $x \in \mathcal{X}$.

Invert$_{sk,x}$: $\mathcal{V} \longrightarrow \mathcal{U}$: A probabilistic inverter indexed by sk and $x \in \mathcal{X}$.

$u_g =$ HTDF.Eval$^{\text{in}}(g, (x_1, u_1, v_1), \ldots, (x_t, u_t, v_t))$: A deterministic input homomorphic evaluation algorithm. It takes some function g: $\mathcal{X}^t \longrightarrow \mathcal{X}$ as input and values $\{x_i \in \mathcal{X}, u_i \in \mathcal{U}, v_i \in \mathcal{V}\}_{i\in[t]}$ and outputs $u_g \in \mathcal{U}$.

$v_g =$ HTDF.Eval$^{\text{out}}(g, v_1, \ldots, v_t)$: A deterministic output homomorphic evaluation algorithm. It takes some function g: $\mathcal{X}^t \longrightarrow \mathcal{X}$ as input and values $\{v_i \in \mathcal{V}\}_{i\in[t]}$ and outputs $v_g \in \mathcal{V}$.

3.1.1. Correctness of Homomorphic Computation. Let $(pk, sk) \leftarrow$HTDF.KeyGen(1^λ) and g: $\mathcal{X}^t \longrightarrow \mathcal{X}$ 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 $v_i = f_{pk, x_i}(u_i)$ for $i \in [t]$. Set $v_g =$ HTDF.Eval$^{\text{out}}(g, v_1, \ldots, v_t)$, $u_g =$ HTDF.Eval$^{\text{in}}(g, (x_1, u_1, v_1), \ldots, (x_t, u_t, v_t))$. The correctness requirement is established for $u_g \in \mathcal{U}$ and $f_{pk,y}(u_g) = v_g$.

3.1.2. Relaxed Correctness of Leveled HTDFs. In a leveled fully homomorphic scheme, the size of noise carried by each input $u_i \in \mathcal{U}$ is $\beta_i \in \mathbb{Z}$. The initial samples selected from the input-distribution $\mathcal{D}_\mathcal{U}$ carries a small noise of β_0 and the noise size β_g of the evaluated input u_g depends on the noise size β_i of u_i, the indices x_i, and the function g. In fact, if the noise size $\beta_g > \beta_{\max}$, where $\beta_{\max}$ is an extreme value of noise size, the correctness of the scheme cannot be guaranteed. Thus, we ought to pay attention to limiting the types of functions that can be calculated homomorphically. A function g is admissible on indices $(x_1, \ldots, x_t)$ if, $\beta_g \leq \beta_{\max}$ whenever u_i carries noise with size $\beta_i \leq \beta_0$.

3.1.3. Distributional Equivalence of Inversion. In order to illustrate the security of our main construction FHS, we demand the following statistical indistinguishability:

$$(pk, sk, x, u, v) \approx_s (pk, sk, x, u', v'), \tag{5}$$

where $(pk, sk) \leftarrow \text{HTDF.KeyGen}(1^\lambda)$, $x \in \mathcal{X}$, $u \leftarrow \mathcal{D}_{\mathcal{U}}, v = f_{pk,x}(u), v' \xleftarrow{\$} \mathcal{V}, u' \leftarrow Inv_{sk,x}((v')$.

3.1.4. HTDF Security. Gorbunov et al. [2] constructed an existential unforgeability FHS based on the security of claw-freeness HTDF. Here, we want to construct a FHS scheme that is strongly unforgeable like Wang et al. [26] did, so the security of HTDF is required to meet the two characteristics of claw-freeness and collision-resistance. In particular, it should be hard to find $u \neq u' \in \mathcal{U}$ and $x, x' \in \mathcal{X}$ such that $f_{pk,x}(u) = f_{pk,x'}(u')$. Notice that if $x = x'$ holds, then (u, u') is a collision, otherwise a claw. Formally, for any PPT attacker $\mathcal{A}$, the following holds:

$$\Pr\left[\begin{array}{c} f_{pk,x}(u) = f_{pk,x'}(u') \\ u \neq u' \in U, x, x' \in X \end{array} \middle| \begin{array}{c} (pk, sk) \leftarrow \text{HTDF.KeyGen} \\ (u, u', x, x') \leftarrow A\left(1^\lambda, pk\right) \end{array}\right] \leq negl(\lambda). \quad (6)$$

3.2. Construction: Basic Algorithms and Security. It is to be remembered that λ is the security parameter. In order to illustrate the HTDF function concisely, we need to introduce some asymptotic public parameters, as shown below.

3.2.1. Parameters. We define flexible $d \leq \text{poly}(\lambda)$ as the maximum depth of the circuit that supported by homomorphic trapdoor function. Choose an integer $n = \text{poly}(\lambda)$ and a sufficiently large prime $q = q(n)$, and set $\ell = (\log q)$. Set $\beta_0 = O(q^{1/2} \cdot \text{poly}(n)) = O(q^{1/2} \cdot \text{poly}(\lambda)), \beta_{\max} = O(4^d n\ell\beta_0), \beta_{RSIS} = O(n\ell)\beta_{\max} < q$.

3.2.2. Construction of HTDF. Below, we give the basic algorithm of the HTDF function $\mathcal{F}$.

Set $\mathcal{X} = \mathbb{Z}_2, \mathcal{V} = R_q^\ell$ and $\mathcal{U} = \{\vec{u} = (\vec{u}_1, \vec{u}_2) \in R_q^{2\times\ell}: \|\vec{u}\|_\infty \leq \beta_{\max}\}$. Define the distribution $\mathcal{D}_{\mathcal{U}}$ and sample $\vec{u} \leftarrow \text{SamDom}(1^{2\ell}, q)$ as in Lemma 4 so that $\|\vec{u}\|_\infty \leq \beta_0$.

$(pk, sk) \leftarrow \text{HTDF.KeyGen}(1^\lambda)$: Run $(h, td) \leftarrow \text{TrapGen}(1^n, 1^\ell, q)$ and set $pk = h \in R_q$ and $sk = td$.

Define $f_{pk,x}(\vec{u}) \triangleq [h, -1]\vec{u} + x\vec{g} = h\vec{u}_1 - \vec{u}_2 + x\vec{g}$.

Define $\vec{u} \leftarrow Inv_{sk,x}(\vec{v})$ to output $\vec{u} \leftarrow \text{SamePre}(h, \vec{v} - x\vec{g}, td)$.

3.2.3. Distributional Equivalence of Inversion. Let $x \in \mathcal{X}$ and $(pk = h, sk = td) \leftarrow HTDF.KeyGen(1^\lambda)$. Let $\vec{u} \leftarrow \mathcal{D}_{\mathcal{U}}$, $\vec{v} = f_{pk,x}(\vec{u}) = [h, -1]\vec{u} + x\vec{g} = h\vec{u}_1 - \vec{u}_2 + x\vec{g}$, $\vec{v}' \xleftarrow{\$} \mathcal{V}, \vec{u}' \leftarrow \text{SamPre}(h, \vec{v}' - x\vec{g}, td)$. Obviously $(\vec{v}' - x\vec{g})$ is uniformly random, then by Lemma 4 and a simple hybrid argument, we can obtain as follows:

$$\left(h, td, \vec{u}, h\vec{u}_1 - \vec{u}_2\right) \approx_s \left(h, td, \vec{u}', \vec{v}' - x\vec{g}\right) \quad (7)$$

We perform the following operations on both sides of the equation: put in an $x \in \mathcal{X}$ and then add $x\vec{g}$ to the last entry to both sides, we can obtain as follows:

$$\left(h, td, x, \vec{u}, \vec{v} = h\vec{u}_1 - \vec{u}_2 + x\vec{g}\right) \approx_s \left(h, td, x, \vec{u}', \vec{v}'\right) \quad (8)$$

3.2.4. HTDF Security. Next, it is proved that if the RSIS assumption holds, the HTDF function $\mathcal{F}$ constructed above is safe. Recall that in the original HTDF security experiment, we generate the public key h along with the secret key td for $h \xleftarrow{\$} R_q^*$.

Theorem 2. *Suppose that the $RSIS_{q,2,\beta_{RSIS}}$-assumption holds for the described parameters, then the HTDF function $\mathcal{F}$ satisfies the HTDF security.*

Proof. Assume that there exists a PPT adversary $\mathcal{A}$, which can have a nonnegligible probability δ win the HTDF security experiment. Then, we establish a probability polynomial time simulator $\mathcal{S}$ that breaks the RSIS $q, 2, \beta_{\text{RSIS}}$ assumption with nonnegligible probability.

Next, we revise the HTDF security experiment, we sample $h \xleftarrow{\$} R_q^*$, instead of sampling $(h, td) \leftarrow \text{TrapGen}(1^n, 1^\ell, q)$ and setting $pk = h$ and $sk = td$. Please note that $sk = td$ has never been used anywhere in the primeval HTDF security experiment. Hence, adversary $\mathcal{A}'$ s point of view among the original HTDF security experiment and the simulated HTDF security experiment are indistinguishable by Lemma 4. Specifically, adversary $\mathcal{A}$ attacks the security experiment of simulated HTDF and gains the security experiment with a probability of at least $\delta - \text{negl}(\lambda)$.

At present, we have proved that the PPT adversary $\mathcal{A}$, who won in the simulated HTDF security experiment, can be used to settle the RSIS problem. Suppose that the successful adversary $\mathcal{A}$ outputs values $\vec{u} \neq \vec{u}' \in \mathcal{U}, x, x' \in \mathcal{X}$ such that $f_{pk,x}(\vec{u}) = f_{pk,x'}(\vec{u}')$. Let $\vec{u}^* = \vec{u} - \vec{u}'$ and $x^* = x' - x$. Then, we obtain as follows:

$$\begin{aligned} f_{pk,x}(\vec{u}) &= [h, -1]\vec{u} + x\vec{g} = [h, -1]\vec{u}' + x'\vec{g} \\ &= f_{pk,x'}(\vec{u}') \Rightarrow [h, -1]\vec{u}^* = x^*\vec{g}. \end{aligned} \quad (9)$$

Additionally, because $\vec{u}, \vec{u}' \in \mathcal{U}$, it holds that $\|\vec{u}\|_\infty, \|\vec{u}'\|_\infty \leq \beta_{\max}$ and thus $\|\vec{u}^*\|_\infty \leq 2\beta_{\max}$. Moreover, since $\vec{u} \neq \vec{u}'$, it holds that $\vec{u}^* \neq \vec{0}$.

To solve the RSIS problem defined by $[h, -1]$, where $h \in R_q^*$, then we talk over the two situations given below:

$x = x'$: In this situation, the small $\vec{u}^* = \vec{u} - \vec{u}' \neq \vec{0}$ is a nonzero solution of the RSIS problem, as $[h, -1](\vec{u} - \vec{u}') = [h, -1]\vec{u}^* = x^*\vec{g} = (x - x')\vec{g} = 0$, .

$x \neq x'$: On this event, we prove that the simulator $\mathcal{S}$ pretty good at using the knowledge of a small $\vec{u}^* \neq \vec{0}$

and some $x^* \neq 0$ satisfying the right side of the (9) to find a nonzero solution of the RSIS problem (similarly as [2]).

Sample $\overrightarrow{t} \overset{\$}{\leftarrow} R_2^2$ and let $\overrightarrow{r} \triangleq [h,-1]\overrightarrow{t} \in R_q$. Calculate $\overrightarrow{t}' = \overrightarrow{g}^{-1}(\overrightarrow{r}/x^*) \in R_2^\ell$ such that $x^*\overrightarrow{g}\,\overrightarrow{t}' = \overrightarrow{r}$. Therefore,

$$\begin{aligned}[h,-1]\left(\overrightarrow{u}^*\overrightarrow{t}' - \overrightarrow{t}\right) &= \left([h,-1]\overrightarrow{u}^*\right)\overrightarrow{t}' - [h,-1]\overrightarrow{t}\\ &= x^*\overrightarrow{g}\,\overrightarrow{t}' - \overrightarrow{r} = \overrightarrow{r} - \overrightarrow{r} = \overrightarrow{0}.\end{aligned} \tag{10}$$

Setting $\overrightarrow{u}^\diamond \triangleq \overrightarrow{u}^*\overrightarrow{t}' - \overrightarrow{t}$, we then have $[h,-1]\overrightarrow{u}^\diamond = \overrightarrow{0}$ and $\|\mathbf{u}^\diamond\|_\infty \leq 2n\ell\beta_{\max} + 1 \leq \beta_{SIS}$. Below the $\overrightarrow{u}^\diamond \neq \overrightarrow{0}$, i.e., $\overrightarrow{t} \neq \overrightarrow{u}^*\overrightarrow{t}'$ have to be proved. Here we show that, even given $(h, \overrightarrow{u}^*, x^*)$, the following inequality holds with overwhelming probability for uniformly random $\overrightarrow{t}$ Actually, we have

$$\widetilde{\mathbf{H}}_\infty\left(\overrightarrow{t}\,|\,\overrightarrow{t}'\right) \geq \widetilde{\mathbf{H}}_\infty\left(\overrightarrow{t}\,|\,[h,-1]\overrightarrow{t}\right) \geq \omega(n), \tag{11}$$

The reason for the first inequality is that $\overrightarrow{t}'$ is completely determined by $\overrightarrow{r} = [h,-1]\overrightarrow{t}$ as $\overrightarrow{g}^{-1}(\cdot)$ is deterministic, and the reason for the second inequality comes from Lemma 4. Therefore, $\Pr[\overrightarrow{t} = \overrightarrow{u}^*\overrightarrow{t}'] \leq 2^{-\omega(n)} = \mathrm{negl}(\lambda)$.

So, if the adversary $\mathcal{A}$ wins any situation in the abovementioned simulated HTDF security game with a nonnegligible probability $\delta/2 - \mathrm{negl}(\lambda)$, the simulator $\mathcal{S}$ will generate an effective and small enough solution to solve the RSIS problem with a probability of $\delta/2 - \mathrm{negl}(\lambda)$. The process of the proof ends. □

3.3. Basic Homomorphic Evaluation. In an HTDF scheme, in order to verify an evaluated function value, we demand to design two corresponding algorithms: one is the input algorithm, another kind is the output algorithm, and both algorithms are deterministic. This is different from a homomorphic encryption system; in a homomorphic encryption system, only one (random or deterministic) homomorphic algorithm that acts on the ciphertext needs to be designed.

The two basic homomorphic algorithms addition and multiplication are described as follows. Here, we will simplify some notations (e.g., $\mathrm{Eval}^{\mathrm{in}}$ instead of $\mathrm{HTDF.Eval}^{\mathrm{in}}$). Recall that for $i = 1, 2$, $\overrightarrow{v}_i = [h,-1]\overrightarrow{u}_i + x_i\overrightarrow{g}$. Let $\|\overrightarrow{u}_i\|_\infty \leq \beta_i$ and $x_i \in \{0,1\}$.

3.3.1. Homomorphic Addition Algorithms. They calculate the sum of the corresponding input and output vectors, respectively.

$\mathrm{Eval}^{\mathrm{in}}(\mathrm{Add}, (x_1, \overrightarrow{u}_1, \overrightarrow{v}_1), (x_2, \overrightarrow{u}_2, \overrightarrow{v}_2)) \triangleq \overrightarrow{u}_1 + \overrightarrow{u}_2 \bmod q$

$\mathrm{Eval}^{\mathrm{out}}(\mathrm{Add}, \overrightarrow{v}_1, \overrightarrow{v}_2) \triangleq \overrightarrow{v}_1 + \overrightarrow{v}_2 \bmod q$

The noise of the addition is bounded by $\beta_1 + \beta_2$. The correctness comes from $(\overrightarrow{v}_1 + \overrightarrow{v}_2) = [h,-1](\overrightarrow{u}_1 + \overrightarrow{u}_2) + (x_1 + x_2)\overrightarrow{g}$.

3.3.2. Homomorphic Multiplication Algorithms. The homomorphic output multiplication algorithm is a multiplication of the output vectors which is performed the multiplication defined in Section 3.1. The homomorphic input multiplication algorithm is asymmetric and it needs partial output, index, and the whole input to compute.

$\mathrm{Eval}^{\mathrm{in}}(Mult, (x_1, \overrightarrow{u}_1, \overrightarrow{v}_1), (x_2, \overrightarrow{u}_2, \overrightarrow{v}_2)) \triangleq \overrightarrow{u}_1 \cdot \overrightarrow{g}^{-1}(\overrightarrow{v}_2) + x_1\overrightarrow{u}_2 \bmod q$

$\mathrm{Eval}^{\mathrm{out}}(Mult, \overrightarrow{v}_1, \overrightarrow{v}_2) \triangleq \overrightarrow{v}_1 \cdot \overrightarrow{g}^{-1}(\overrightarrow{v}_2) \bmod q$

The noise of multiplication is bounded by $n\ell\beta_1 + |x_1|\beta_2 = n\ell\beta_1 + \beta_2$. Assuming $\overrightarrow{v}_i = [h,-1]\overrightarrow{u}_i + x_i\overrightarrow{g}$, the correctness follows by the computation:

$$\begin{aligned}\overrightarrow{v}_1 \cdot \overrightarrow{g}^{-1}(\overrightarrow{v}_2) &= ([h,-1]\overrightarrow{u}_1 + x_1\overrightarrow{g}) \cdot \overrightarrow{g}^{-1}(\overrightarrow{v}_2)\\ &= [h,-1]\overrightarrow{u}_1 \cdot \overrightarrow{g}^{-1}(\overrightarrow{v}_2)\\ &\quad + x_1([h,-1]\overrightarrow{u}_2 + x_2\overrightarrow{g})\\ &= [h,-1]\left(\overrightarrow{u}_1 \cdot \overrightarrow{g}^{-1}(\overrightarrow{v}_2) + x_1\overrightarrow{u}_2\right) + x_1x_2\overrightarrow{g}.\end{aligned} \tag{12}$$

3.4. Homomorphic Output and Input Evaluation. According to the abovementioned discussion, it is very easy to homomorphically calculate an arithmetic circuit or Boolean circuit similar as [2], we are not satisfied. We want to show how to design the scheme as well as [26] based on the fact that the noise growth is asymmetric.

3.4.1. Homomorphic Output Evaluation. Below we describe the definition of the homomorphic output evaluation algorithm

$$\mathrm{Eval}^{\mathrm{out}}\left(\Pi, \overrightarrow{v}_0, \left\{\overrightarrow{v}_{0,i}\right\}_{i\in[w]}, \left\{\overrightarrow{v}_j\right\}_{j\in[t]}\right) \longrightarrow \overrightarrow{v}_\Pi, \tag{13}$$

for a length is L and a width is w permutation branching program Π. In the initialization stage, we will assign $\overrightarrow{v}_0$, $\left\{\overrightarrow{v}_{0,i}\right\}_{i\in[w]}$ below and $\overrightarrow{v}_j$ is the vector such that $\overrightarrow{v}_j = [h,-1]\overrightarrow{u}_j + x_j\overrightarrow{g}$, where $\overrightarrow{u}_j = (\overrightarrow{u}_{j,1}, \overrightarrow{u}_{j,2})$. Recall that $\{(r(k), \xi_{k,i,0}, \xi_{k,i,1})\}_{k\in[L],i\in[w]}$ is a description of the permutation branching program Π, and the initial state vector is defined as the first w-dimensional unit vector $\overrightarrow{e}_0 = (1, 0, 0, \ldots, 0)$. For $k \in [L]$ and $i \in [w]$, it holds that:

$$\overrightarrow{e}_k[i] = \overrightarrow{e}_{k-1}\left[\xi_{k,i,1}\right] \cdot x_{r(k)} + \overrightarrow{e}_{k-1}\left[\xi_{k,i,0}\right] \cdot \left(1 - x_{r(k)}\right). \tag{14}$$

Here, we give the full description of the process of the homomorphic output evaluation algorithm $\mathrm{Eval}^{\mathrm{out}}$ of the permutation branching program Π.

Initialization: Let $\overrightarrow{v}_{k,i}$ be an output relevant to the state $\overrightarrow{e}_k[i]$, where $k \in [L], i \in [w]$,

(1) Choose $\overrightarrow{v}_0 \overset{\$}{\leftarrow} R_q^\ell$ and set it to be a corresponding output of the state 1.
(2) Choose $\overrightarrow{v}_{0,i} \overset{\$}{\leftarrow} R_q^\ell$ and set it to be a corresponding initial output of the initial state $\overrightarrow{e}_0[i]$.
(3) Set $\overrightarrow{v}_j^{\,\prime} \triangleq \overrightarrow{v}_0 - \overrightarrow{v}_j$, where $\overrightarrow{v}_j$ (such that $\overrightarrow{v}_j = [h,-1]\overrightarrow{u}_j + x_j\overrightarrow{g}$) is an output corresponding to x_j, and set it to be a corresponding output of $(1 - x_j)$.

Calculation: For $k = 1, 2, \ldots, L$, suppose that at step $k-1$, we have $\{\vec{v}_{k-1,i}\}_{i\in[w]}$. Then, we can calculate inductively as follows:

$$\vec{v}_{k,i} = \vec{v}_{r(k)} \cdot \vec{g}^{-1}\left(\vec{v}_{k-1,\xi_{k,i,1}}\right) + \overline{\vec{v}}_{r(k)} \cdot \vec{g}^{-1}\left(\vec{v}_{k-1,\xi_{k,i,0}}\right). \tag{15}$$

Final output: After finishing the computation, we have the value of $\{\vec{v}_{L,i}\}_{i\in[w]}$. Finally, output the final output $\vec{v}_{L,1}$ corresponding to the state $\vec{e}_L[1]$, i.e., $\vec{v}_\Pi = \vec{v}_{L,1}$.

3.4.2. Homomorphic Input Evaluation. Here, we give the full description of the process of the homomorphic input evaluation algorithm $\text{Eval}^{\text{in}}(\Pi, (1, \vec{u}_0, \vec{v}_0)), \{(\vec{e}_0[i], \vec{u}_{0,i}, \vec{v}_{0,i})\}_{i\in[w]}, \{(x_j, \vec{u}_j, \vec{v}_j)\}_{j\in[t]}) \longrightarrow \vec{u}_\Pi$ of the permutation branching program Π.

Initialization: Let $\vec{u}_{k,i}$ be an input relevant to the state $\vec{e}_k[i]$, where $k \in [L], i \in [w]$.

(1) Sample $\vec{u}_0 = (\vec{u}_{0,1}, \vec{u}_{0,2}) \leftarrow \mathcal{D}_\mathcal{U}$ (such that $\vec{v}_0 = [h, -1]\vec{u}_0 + 1 \cdot \vec{g}$) and set it as input relevant to the state 1.

(2) Sample $\vec{u}_{0,i} = (\vec{u}_{0,i,1}, \vec{u}_{0,i,2}) \leftarrow \mathcal{D}_\mathcal{U}$ (such that $\vec{v}_{0,i} = [h, -1]\vec{u}_{0,i} + \vec{e}_0[i]\vec{g}$) and set it as the initial input relevant to the initial state $\vec{e}_0[i]$.

(3) Set $\overline{\vec{u}}_j \triangleq \vec{u}_0 - \vec{u}_j$, where $\vec{u}_j = (\vec{u}_{j,1}, \vec{u}_{j,2})$ (such that $\vec{v}_j = [h, -1]\vec{u}_j + x_j\vec{g}$) is an input corresponding to x_j, and set it to the input relevant to $(1 - x_j)$.

Calculation: For $k = 1, 2, \ldots, L$, suppose that at step $k-1$, we have $\{\vec{u}_{k-1,i}\}_{i\in[w]}$. Then, we can calculate inductively as follows.

$$\vec{u}_{k,i} = \left(\vec{u}_{r(k)} \cdot \vec{g}^{-1}\left(\vec{v}_{k-1,\xi_{k,i,1}}\right) + x_{r(k)} \cdot \vec{u}_{k-1,\xi_{k,i,1}}\right) + \left(\overline{\vec{u}}_{r(k)} \cdot \vec{g}^{-1}\left(\vec{v}_{k-1,\xi_{k,i,0}}\right) + \left(1 - x_{r(k)}\right) \cdot \vec{u}_{k-1,\xi_{k,i,0}}\right). \tag{16}$$

Final input: After finishing the computation process, we have the value of $\{\vec{u}_{L,i}\}_{i\in[w]}$. Finally, output the final input $\vec{u}_{L,1}$ corresponding to $\vec{e}_L[1]$, i.e., $\vec{u}_\Pi = \vec{u}_{L,1}$.

3.5. Correctness of Homomorphic Evaluation and Noise Analysis. In the following lemmas, we testify the validity of homomorphic input and homomorphic output algorithms and explain in detail the noise growth in the above homomorphic evaluation.

Lemma 5. *Suppose that* $\text{Eval}^{out}(\Pi, \vec{v}_0, \{\vec{v}_{0,i}\}_{i\in[w]}, \{\vec{v}_j\}_{j\in[t]}) \longrightarrow \vec{v}_\Pi$ *and* $\text{Eval}^{in}(\Pi, (1, \vec{u}_0, \vec{v}_0), \{(\vec{e}_0[i], \vec{u}_{0,i}, \vec{v}_{0,i})\}_{i\in[w]}, \{(x_j, \vec{u}_j, \vec{v}_j)\}_{j\in[t]}) \longrightarrow \vec{u}_\Pi$, *where* $\vec{v}_0 = [h, -1]\vec{u}_0 + 1 \cdot \vec{g}$, $\vec{v}_{0,i} = [h, -1]\vec{u}_{0,i} + \vec{e}_0[i]\vec{g}$ *and* $\vec{v}_j = [h, -1]\vec{u}_j + x_j\vec{g}$ *for* $i \in [w]$, $j \in [t]$. *Then for all* $k \in [L], i \in [w]$, *we have* $\vec{v}_{k,i} = [h, -1]\vec{u}_{k,i} + \vec{e}_k[i]\vec{g}$. *For* $k = L$, *we have* $\vec{v}_{L,1} = [h, -1]\vec{u}_{L,1} + \vec{e}_L[1]\vec{g}$.

Proof. Using formulas (1), (4), and (5) and the above conditions, for all $k \in [L], i \in [w]$, it holds as follows:

$$\begin{aligned}
\vec{v}_{k,i} &= \vec{v}_{r(k)} \cdot \vec{g}^{-1}\left(\vec{v}_{k-1,\xi_{k,i,1}}\right) + \overline{\vec{v}}_{r(k)} \cdot \vec{g}^{-1}\left(\vec{v}_{k-1,\xi_{k,i,0}}\right) \\
&= \left([h, -1]\vec{u}_{r(k)} + x_{r(k)}\vec{g}\right) \cdot \vec{g}^{-1}\left(\vec{v}_{k-1,\xi_{k,i,1}}\right) + \left([h, -1]\overline{\vec{u}}_{r(k)} + \left(1 - x_{r(k)}\right)\vec{g}\right) \cdot \vec{g}^{-1}\left(\vec{v}_{k-1,\xi_{k,i,0}}\right) \\
&= [h, -1]\vec{u}_{r(k)} \cdot \vec{g}^{-1}\left(\vec{v}_{k-1,\xi_{k,i,1}}\right) + x_{r(k)}[h, -1]\vec{u}_{k-1,\xi_{k,i,1}} + x_{r(k)} \cdot \vec{e}_{k-1}\left[\xi_{k,i,1}\right]\vec{g} \\
&\quad + [h, -1]\overline{\vec{u}}_{r(k)} \cdot \vec{g}^{-1}\left(\vec{v}_{k-1,\xi_{k,i,0}}\right) + \left(\left(1 - x_{r(k)}\right)[h, -1]\vec{u}_{k-1,\xi_{k,i,0}}\right. \\
&\quad + \left(1 - x_{r((k))}\right) \cdot \vec{e}_{k-1}\left[\xi_{k,i,0}\right]\vec{g} \\
&= [h, -1]\left[\left(\vec{u}_{r(k)} \cdot \vec{g}^{-1}\left(\vec{v}_{k-1,\xi_{k,i,1}}\right) + x_{r(k)} \cdot \vec{u}_{k-1,\xi_{k,i,1}}\right) + \left(\left(\overline{\vec{u}}_{r(k)} \cdot \vec{g}^{-1}\left(\vec{v}_{k-1,\xi_{k,i,0}}\right)\right) + \left(1 - x_{r(k)}\right) \cdot \vec{u}_{k-1,\xi_{k,i,0}}\right)\right] \\
&\quad + \left[x_{r(k)} \cdot \vec{e}_{k-1}\left[\xi_{k,i,1}\right] + \left(1 - x_{r(k)}\right) \cdot \vec{e}_{k-1}\left[\xi_{k,i,0}\right]\right] \cdot \vec{g} = [h, -1]\vec{u}_{k,i} + \vec{e}_k[i]\vec{g}.
\end{aligned} \tag{17}$$

The process of proof ends. □

Lemma 6. *Assuming that* $\text{Eval}^{in}(\Pi, (1, \vec{u}_0, \vec{v}_0), \{(\vec{e}_0[i], \vec{u}_{0,i}, \vec{v}_{0,i})\}_{i\in[w]}, \{(x_j, \vec{u}_j, \vec{v}_j)\}_{j\in[t]}) \longrightarrow \vec{u}_\Pi$ *and all the noises of the inputs are bounded by* β, *i.e.,* $\|\vec{u}_0\|_\infty, \|\vec{u}_{0,i}\|_\infty, \|\vec{u}_j\|_\infty \leq \beta$, *then we have* $\|\vec{u}_\Pi\|_\infty \leq 3n\ell L\beta + \beta$.

Proof. We use the inductive method to prove the lemma. That is, for any step $k = 0, 1, 2, \ldots, L$ and $i \in [w]$, we will show that $\|\vec{u}_{k,i}\|_\infty \leq 3n\ell k\beta + \beta$.

If $k = 0$, it is not difficult to find that all the initial noises are such that $\|\vec{u}_{0,i}\|_\infty \leq \beta, i \in [w]$.

Suppose that at step $k-1$, we have $\|\vec{u}_{k-1,i}\|_\infty \leq 3n\ell(k-1)\beta + \beta$. Then, according to formula (16), we have as follows:

$$\begin{aligned}
\|\vec{u}_{k,i}\|_\infty &= \Big\|\Big(\vec{u}_{r(k)}\cdot\vec{g}^{-1}\big(\vec{v}_{k-1,\xi_{k,i,1}}\big)+x_{r(k)}\cdot\vec{u}_{k-1,\xi_{k,i,1}}\Big) \\
&\quad +\Big(\overline{\vec{u}}_{r(k)}\cdot\vec{g}^{-1}\big(\vec{v}_{k-1,\xi_{k,i,0}}\big)+\big(1-x_{r(k)}\big)\cdot\vec{u}_{k-1,\xi_{k,i,0}}\Big)\Big\|_\infty \\
&\le \Big\|\vec{u}_{r(k)}\cdot\vec{g}^{-1}\big(\vec{v}_{k-1,\xi_{k,i,1}}\big)\Big\|_\infty \\
&\quad +\Big\|x_{r(k)}\cdot\vec{u}_{k-1,\xi_{k,i,1}}\Big\|_\infty+\Big\|\overline{\vec{u}}_{r(k)}\cdot\vec{g}^{-1}\big(\vec{v}_{k-1,\xi_{k,i,0}}\big)_\infty+\big(1-x_{r((k))}\big)\cdot\vec{u}_{k-1,\xi_{k,i,0}}\Big\|_\infty \\
&\le n\ell\beta+x_{r(k)}\cdot(3n\ell(k-1)\beta+\beta) \\
&\quad +2n\ell\beta+\big(1-x_{r(k)}\big)\cdot(3n\ell(k-1)\beta+\beta) \\
&= 3n\ell k\beta+\beta,
\end{aligned} \tag{18}$$

where $\|\overline{\vec{u}}_{r(k)}\|_\infty = \|\vec{u}_0-\vec{u}_{r(k)}\|_\infty \le \|\vec{u}_0\|_\infty + \|\vec{u}_{r(k)}\|_\infty \le \beta+\beta = 2\beta$.

Through induction, we obtain $\|\vec{u}_\Pi\|_\infty = \|\vec{u}_{L,1}\|_\infty \le 3n\ell L\beta+\beta$. The process of proof ends. □

Remark 1. Barrington's theorem [32] states that a circuit with a depth of d can be transformed into a permutation branching program with a length of 4^d. Thus, simply by setting $d \le \text{poly}(\lambda)$, the resulting maximum noise level is less than $O(4dn\ell\beta)$. In particular, in the setting where it only requires $d = O(\log\lambda)$, we obtain the modulus $q = \text{poly}(\lambda) > O(4^d n\ell\beta)$ to be a middle large polynomial, which can supply a better security for the scheme. In other words, it is based on Ideal-SVP assumption with polynomial approximation factors.

4. Strongly Unforgeable NTRU-Based FHS

We construct a single key FHS scheme that is strongly unforgeable against selective chosen-message-attack (SU-sCMA) using the HTDF function, which appears in the form of a black box in the fore chapter.

4.1. Definition. A single data-set homomorphic signature scheme consists of the following seven PPT algorithms PrmsGen, KeyGen, Sign, SignEval, Process, and Verify:

prms←PrmsGen$(1^\lambda, 1^N)$: Input the security parameter λ and the maximum data-size N, and output public parameters prms. In addition, the security parameters define the message space $\mathcal{X}$.

(pk, sk)←KeyGen(1^λ): Take the security parameter λ as input. Output the key pair (pk, sk) of verification key and signing key.

$(\sigma_1, \ldots, \sigma_N)$←Sign$_{sk}(prms, x_1, \ldots, x_N)$: Sign data $(x_1, \ldots, x_N) \in \mathcal{X}^N$.

$\sigma_g = \text{SignEval}_{\text{prms}}(g, (x_1, \sigma_1), \ldots, (x_t, \sigma_t))$: Deterministically and homomorphically evaluate a signature σ_g for some function g over $(x_1, \ldots, x_t) \in \mathcal{X}^t$.

$v_g = \text{Process}_{\text{prms}}(g)$: A deterministic algorithm that can homomorphically evaluate a certificate v_g from the public parameters prms.

Verify$_{pk}(v_g, y, \sigma_g)$: Verify that y is indeed the output of function g by proving that σ_g and v_g are corresponding.

4.1.1. Correctness. For prms←PrmsGen$(1^\lambda, 1^N)$, (pk, sk)←KeyGen$(1^\lambda, prms)$, $(x_1, \ldots, x_N) \in \mathcal{X}^N$, $(\sigma_1, \ldots, \sigma_N)$←Sign$_{sk}$ (prms, $x_1, \ldots, x_N$), and $g\colon \mathcal{X}^N \longrightarrow \mathcal{X}$, we need the following equation:

$$\text{Verify}_{pk}\big(v_g, y = g(x_1, \ldots, x_N), \sigma_g\big) = \text{accept}, \tag{19}$$

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

Notice that the correctness of singing by setting $g = id_i$ and hence omitted, where id_i is the i-th identity mapping.

4.1.2. Relaxed Correctness of Leveled FHS. Since the relaxed correctness of the leveled FHS can be deduced from the correctness of the leveled HTDF, we omit this process.

4.1.3. Security Experiment. The SU-sCMA security experiment for FHS describes the adversary needs to determine the data to be signed before receiving public parameters and verification key. Then, the adversary struggled to discover (g, y', σ') so that it could meet the conditions for winning the experiment. It is noted that here we do not require $y = y'$ or $y \neq y'$. Therefore, if $y = y'$ holds, then σ' is a strongly forgeable signature, otherwise it is an existentially forgeable signature.

We now proceed to define the SU-sCMA security experiment for FHS among an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows:

The adversary $\mathcal{A}$ selects at one-time data $(x_1, \ldots, x_N) \in \mathcal{X}^N$ and sends it to the challenger $\mathcal{C}$.

The challenger $\mathcal{C}$ runs prms←PrmsGen$(1^\lambda, 1^N)$ and (pk, sk)←$KeyGen(1^\lambda, prms)$ and returns $(prms, pk)$ back to the adversary $\mathcal{A}$.

The challenger $\mathcal{C}$ runs $(\sigma_1, \ldots, \sigma_N)$←Sign$_{\text{sk}}(prms, x_1, \ldots, x_N)$ and sends the signatures $(\sigma_1, \ldots, \sigma_N)$ to the adversary $\mathcal{A}$.

The adversary $\mathcal{A}$ selects a function $g\colon \mathcal{X}^N \longrightarrow \mathcal{X}$ and values y', σ'. Let $y = g(x_1 \ldots, x_N)$. If all the following conditions are met, $\mathcal{A}$ wins:

(1) g is admissible on the data-set $(x_1, \ldots, x_N)$;
(2) $\sigma' \neq \sigma_g$, where $\sigma_g = \mathrm{SignEval}_{\mathrm{prms}}(g, (x_1, \sigma_1), \ldots, (x_N, \sigma_N))$;
(3) $\mathrm{Verify}_{pk}(v_g, y', \sigma')$ accept, where $v_g = \mathrm{Process}_{\mathrm{prms}}(g)$.

If for all PPT adversary $\mathscr{A}$, inequality $\Pr[\mathscr{A} \text{wins}] \leq \mathrm{negl}(\lambda)$ always holds. We say that the FHS scheme is SU-sCMA secure.

Remark 2. Define the stronger SU-aCMA security notion for FHS describing the strongly unforgeable adaptive chosen-message-attack game, where the adversary can select data to be signed after seeing the public parameters and the verification key. It is to be noted that an adaptively secure FHS can be transformed from a selectively secure FHS together with a chameleon hash with homomorphism [2], and that our HTDF built in the previous section is a chameleon hash. Therefore, we believe that we can obtain an adaptively secure NTRU-based FHS via the transformation.

4.2. Construction. Let $\mathscr{F} = (\mathrm{HTDF.KeyGen}, \mathrm{f}, \mathrm{Inv}, \mathrm{HTDF.Eval}^{\mathrm{in}}, (\mathrm{HTDF.Eval}^{\mathrm{out}})$ be an HTDF with index space $\mathscr{X}$, input space $\mathscr{U}$, output space $\mathscr{V}$ and some efficiently sampleable input distribution $\mathscr{D}_{\mathscr{U}}$ over $\mathscr{U}$. We construct an FHS scheme $\mathscr{S} = (\mathrm{PrmsGen}, \mathrm{KeyGen}, \mathrm{Sign}, \mathrm{SignEval}, \mathrm{Process}, \mathrm{Verify})$, with message space $\mathscr{X}$ as follows.

$\mathrm{prms} \leftarrow \mathrm{PrmsGen}(1^\lambda, 1^N)$: Sample $v_i \leftarrow \mathscr{V}, i \in [N]$ and set public parameters $\mathrm{prms} = (v_1, \ldots, v_N)$.

$(pk, sk) \leftarrow \mathrm{KeyGen}(1^\lambda, prms)$: Choose $(pk', sk') \leftarrow \mathrm{HTDF.KeyGen}$ and set $pk = pk', sk = sk'$.

$(\sigma_1, \ldots, \sigma_N) \leftarrow Sign_{sk}(\mathrm{prms}, x_1, \ldots, x_N)$: Sample $u_i \leftarrow Inv_{sk', x_i}(v_i)$ and set $\sigma_i = u_i, i \in [N]$.

$\sigma_g = SignEval_{\mathrm{prms}}(g, (x_1, \sigma_1), \ldots, (x_t, \sigma_t))$: Run $\mathrm{HTDF.Eval}^{\mathrm{in}}(g), (x_1, u_1, v_1), \ldots, (x_t, u_t, v_t))$ and output u_g, then set $\sigma_g = u_g$.

$v_g = \mathrm{Process}_{\mathrm{prms}}(g)$: Run $\mathrm{HTDF.Eval}^{\mathrm{out}}(g, v_1, \ldots, v_t)$ and output the result v_g.

$Verify_{pk}(v_g, y, \sigma_g)$: If $f_{pk', y}(\sigma_g) = v_g$ accept, else reject.

4.2.1. Correctness. The discussion on the correctness of the above-built leveled FHS is the same as the discussion on the correctness of the underlying leveled HTDF $\mathscr{F}$, so it is omitted.

4.2.2. Security. Now, we testify that the above FHS scheme is SU-sCMA secure.

Theorem 3. *Suppose $\mathscr{F}$ is a secure HTDF function, then the above FHS scheme $\mathscr{S}$ is SU-sCMA secure.*

Proof. Assume there exists a PPT adversary $\mathscr{A}$ that wins the SU-sCMA security experiment of FHS with nonnegligible probability δ. We build a PPT reduction $\mathscr{B}$ that breaks the HTDF security of $\mathscr{F}$.

In the SU-sCMA experiment, adversary $\mathscr{A}$ first selects a data $(x_1, \ldots, x_N) \in \mathscr{X}^N$ and then receives $\mathrm{prms} = (v_1, \ldots, v_N)$, pk' and $(\sigma_1, \ldots, \sigma_N)$, where $v_i \leftarrow \mathscr{V}, (pk', sk') \leftarrow \mathrm{HTDF.KeyGen}$ and $\sigma_i = u_i \leftarrow Inv_{sk', x_i}(v_i)$. Next, we revise the experiment by sampling $u_i \leftarrow \mathscr{D}_{\mathscr{U}}$ and setting $v_i = f_{pk', x_i}(u_i)$. Adversary $\mathscr{A}'$ s point of view among the original security experiment and the changed security experiment are indistinguishable by the distributional equivalence of inversion property of the underlying HTDF function. Therefore, $\mathscr{A}$ attacks the changed experiment and wins with a probability of at least $\delta - \mathrm{negl}(\lambda)$.

Then, we prove that there is a PPT reduction $\mathscr{B}$, which can take any PPT adversary $\mathscr{A}$, and the adversary has the ability to win the changed experiment with nonnegligible advantage $\delta - \mathrm{negl}(\lambda)$. Hence, the probability of breaking the HTDF security experiment of the underlying $\mathscr{F}$ is $\delta - \mathrm{negl}(\lambda)$.

After receiving a challenge public key pk' and a data-set $(x_1, \ldots, x_N)$, the reduction $\mathscr{B}$ selects $\{\sigma_i = u_i\}_{i \in [N]}$ and computes $\{v_i\}_{i \in [N]}$ as in the changed experiment and returns $(pk', \{\sigma_i, v_i\}_{i \in [N]})$ back to $\mathscr{A}$.

Suppose that the adversary $\mathscr{A}$ winning the changed experiment outputs values (g, y', σ'), where $g: \mathscr{X}^N \longrightarrow \mathscr{X}$ is an admissible function and $\sigma' = u'$. Let $y = g(x_1, \ldots, x_N), u_g = \sigma_g = \mathrm{SignEval}_{\mathrm{prms}}(g, (x_1, \sigma_1), \ldots, ((x_t, \sigma_t)), v_g = \mathrm{Process}_{\mathrm{prms}}(g)$. So, on the one hand, $f_{pk, y'}(u') = v_g$ holds as the forged signature σ' verifies. On the other hand, $f_{pk, y}(u_g) = v_g$ also holds as g is admissible. Thus, we have values $u_g \neq u' \in \mathscr{U}$ and $y, y' \in \mathscr{X}$ satisfying $f_{pk, y}(u_g) = f_{pk, y'}(u')$, which allows $\mathscr{B}$ to break HTDF security of $\mathscr{F}$ with probability $\delta - \mathrm{negl}(\lambda)$ whenever $\mathscr{A}$ wins the changed experiment with probability $\delta - \mathrm{negl}(\lambda)$. □

5. Conclusion and Open Problem

In this paper, we propose an SU-sCMA secure NTRU-based FHS scheme, which can be transformed to an SU-aCMA one with the help of NTRU-based HTDF functions (and without additional assumptions) using a similar transformation as introduced in [2]. Our next station is improving the efficiency of our FHS scheme and accomplishing the implementation with the consideration of practical attacks [36–39]. In addition, it is well known that the leveled FHE can be transformed into pure FHE via bootstrapping [40], the only workable way to get pure FHE at the present. However, there is no similar bootstrapping to improve homomorphic capability in the HS scenario, as the evaluated signature not only certifies the corresponding message, but also certifies the computation process. Therefore, it is still open to build a pure FHS.

Data Availability

No data were used in this work.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Nature Science Foundation of China (Nos. 61972124, 11974096, and U1705264), Ningbo 2025 Major Project of Science and Technology Innovation (No. 2021Z109), the Zhejiang Provincial Natural Science Foundation of China (No. LY19F020019), and the Research Foundation of Hangzhou Normal University (No. 2020QDL016).

References

[1] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pp. 169–178, Bethesda, MD, USA, June, 2009.

[2] S. Gorbunov, V. Vaikuntanathan, and D. Wichs, "Leveled fully homomorphic signatures from standard lattices," in *Proceedings of the Forty-Seventh Annual ACM STOC*, pp. 469–477, Portland, OR, USA, June, 2015.

[3] R. L. Rivest, "Two Signature schemes," 2000, http://people.csail.mit.edu/rivest/pubs.html.

[4] R. Johnson, D. Molnar, D. Song, and D. Wagner, "Homomorphic signature schemes," in *Cryptographers Track at the Rsa Conference on Topics in Cryptology*, pp. 244–262, Springer, Berlin Heidelberg, 2002.

[5] F. Zhao, T. Kalkert, M. Medard, and K. J. Han, "Signatures for content distribution with network coding," in *Proceedings of the International Symposium on Information Theory*, pp. 556–560, Nice, France, June, 2007.

[6] D. Charles, K. Jain, and K. Lauter, "Signatures for network coding," *International Journal of Information and Coding Theory*, vol. 1, no. 1, p. 3, 2009.

[7] R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin, "Secure network coding over the integers," in *Proceedings of the International Association for Cryptologic Research*, pp. 142–160, Paris, France, May, 2010.

[8] N. Attrapadung and B. Libert, "Homomorphic network coding signatures in the standard model," *Public Key Cryptography - PKC 2011*, vol. 6571, pp. 17–34, 2011.

[9] D. Boneh and D. M. Freeman, "Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures," *Public Key Cryptography - PKC 2011*, vol. 6571, pp. 1–16, 2011.

[10] X. Hu, S. Zheng, J. Gong, G. Cheng, G. Zhang, and R. Li, "Enabling linearly homomorphic signatures in network coding-based named data networking," *CFI*, vol. 2, pp. 1-2, 2019.

[11] C. J. Lin, R. Xue, S. J. Yang, X. Huang, and S. Li, "Linearly homomorphic signatures from lattices," *The Computer Journal*, vol. 63, no. 12, pp. 1871–1885, 2020.

[12] D. Boneh and D. M. Freeman, "Homomorphic signatures for polynomial functions," *Advances in Cryptology - EUROCRYPT 2011*, vol. 6632, pp. 149–168, 2011.

[13] D. Catalano, D. Fiore, and B. Warinschi, "Homomorphic signatures with efficient verification for polynomial functions," *Advances in Cryptology - CRYPTO 2014*, vol. 8616, pp. 371–389, 2014.

[14] M. Ajtai, "Generating Hard Instances of Lattice Problems," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing STOC*, pp. 99–108, Philadelphia Pennsylvania USA, July, 1996.

[15] D. Micciancio and O. Regev, "Worst-case to average-case reductions based on Gaussian measures," in *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science FOCS*, pp. 372–381, Rome, Italy, October, 2004.

[16] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing STOC*, pp. 197–206, Victoria, British Columbia, Canada, May, 2008.

[17] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert, "Bonsai trees, or how to delegate a lattice basis," *Advances in Cryptology - EUROCRYPT 2010*, vol. 6110, pp. 523–552, 2010.

[18] D. Micciancio and C. Peikert, "Trapdoors for lattices: simpler, tighter, faster, smaller," *Advances in Cryptology - EUROCRYPT 2012*, vol. 7237, pp. 700–718, 2012.

[19] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based," *CRYPTO, LNCS*, vol. 8042, pp. 75–92, 2013.

[20] X. Boyen, X. Fan, and E. Shi, "Adaptively secure fully homomorphic signatures based on lattices," 2014, http://eprint.iacr.org/2014/916.

[21] S. Agrawal, D. Boneh, and X. Boyen, "Efficient lattice (H)IBE in the standard model," *Advances in Cryptology - EUROCRYPT 2010*, vol. 6110, pp. 553–572, 2010.

[22] R. Tsabary, "An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both," *Theory of Cryptography*, vol. 10678, pp. 489–518, 2017.

[23] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures," in *Topics in Cryptology - CT-RSA 2011*, pp. 376–392, Springer, New York, NY, USA, 2011.

[24] F. Luo, F. Wang, K. Wang, and K. Chen, "A more efficient leveled strongly-unforgeable fully homomorphic signature scheme," *Information Sciences*, vol. 480, pp. 70–89, 2019.

[25] D. Apon, X. Fan, and F. Liu, "Vector encoding over lattices and its applications," 2017, https://eprint.iacr.org/2017/455.

[26] F. Wang, K. Wang, B. Li, and Y. Gao, "Leveled strongly-unforgeable identity-based fully homomorphic signatures," *Lecture Notes in Computer Science*, vol. 9290, pp. 42–60, 2015.

[27] C. Wang, B. Wu, and H. Yao, "Leveled adaptively strong-unforgeable identity-based fully homomorphic signatures," *IEEE Access*, vol. 8, no. 99, Article ID 119431, 2020.

[28] Y. Wang and M. Wang, "A New Fully Homomorphic Signatures from Standard Lattices," in *Proceedings of the International Conference on Wireless Algorithms, Systems, and Applications*, Qingdao, China, September, 2020.

[29] J. Hoffstein, J. Pipher, and J. H. Silverman, Ntru, NTRU: a ring-based public key cryptosystem," *Lecture Notes in Computer Science*, vol. 1423, pp. 267–288, 1998.

[30] D. Stehlé and R. Steinfeld, "Making NTRU as secure as worst-case problems over ideal lattices," *Advances in Cryptology - EUROCRYPT 2011*, vol. 6632, pp. 27–47, 2011.

[31] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte, "NTRUSign: digital signatures using the NTRU lattice," in *Topics in Cryptology - CT-RSA 2003*, vol. 2612, pp. 122–140, Springer, New York, NY, USA, 2003.

[32] D. Barrington, "Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1," *Journal of Computer and System Sciences*, vol. 38, pp. 1–5, 1986.

[33] Y. Yu, G. Xu, and X. Wang, "Provably secure NTRU instances over prime cyclotomic rings," in *Proceedings of the Public-Key Cryptography – PKC 2017*, pp. 409–434, Amsterdam, The Netherlands, March, 2017.

[34] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proceedings of the Annual ACM Symposium on Theory of Computing STOC*, pp. 1219–1234, Quebec, Canada, May, 2012.

[35] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory*, vol. 6, no. 3, pp. 1–36, 2014.

[36] M. Albrecht, S. Bai, and L. Ducas, "A subfield lattice attack on overstretched NTRU assumptions," *Advances in Cryptology - CRYPTO 2016*, vol. 9814, pp. 153–178, 2016.

[37] P. Kirchner and P.-A. Fouque, "Revisiting lattice attacks on overstretched NTRU parameters," *Lecture Notes in Computer Science*, vol. 10210, pp. 3–26, 2017.

[38] J. Ding, J. Deaton, K. Schmidt, Vishakha, and Z. Zhang, "A simple and efficient key reuse attack on NTRU cryptosystem," 2019, http://eprint.iacr.org/2019/1022.

[39] M. Albrecht and L. Ducas, "Lattice attacks on NTRU and lwe: a history of refinements," 2021, http://eprint.iacr.org/2021/799.

[40] J. Alperin-Sheriff and C. Peikert, "Faster bootstrapping with polynomial error," *Advances in Cryptology - CRYPTO 2014*, vol. 8618, pp. 297–314, 2014.