Improved Universal Thresholdizer from Threshold Fully Homomorphic Encryption

Jung Hee Cheon[1,2], Wonhee Cho[1], and Jiseung Kim[3]

[1] Seoul National University, Republic of Korea.
{jhcheon,wony0404}@snu.ac.kr
[2] Crypto Lab Inc., Seoul, Republic of Korea.
[3] Jeonbuk National University, Republic of Korea.
{jiseungkim}@jbnu.ac.kr

Abstract. The Universal Thresholdizer (CRYPTO'18) is a cryptographic scheme that facilitates the transformation of any cryptosystem into a threshold cryptosystem, making it a versatile tool for threshold cryptography. For instance, this primitive enables the black-box construction of a one-round threshold signature scheme based on the Learning with Error problem, as well as a one-round threshold chosen ciphertext attack-secure public key encryption, by being combined with non-threshold schemes. The compiler is constructed in a modular fashion and includes a compact threshold fully homomorphic encryption, a non-interactive zero-knowledge proof with preprocessing, and a non-interactive commitment. An instantiation of the Universal Thresholdizer can be achieved through the construction of a compact threshold fully homomorphic encryption. Currently, there are two threshold fully homomorphic encryptions based on linear secret sharing, with one using Shamir's secret sharing and the other using the $\{0,1\}$-linear secret sharing scheme ($\{0,1\}$-LSSS). The former fails to achieve compactness as the size of its ciphertext is $O(N \log N)$, where N is the number of participants in the distributed system. Meanwhile, the latter provides compactness, with a ciphertext size of $O(\log N)$, but requires $O(N^{4.3})$ share keys on each party, leading to high communication costs.
In this paper, we propose a communication-efficient Universal Thresholdizer by revisiting the threshold fully homomorphic encryption. Our scheme reduces the number of share keys required on each party to $O(N^{2+o(1)})$ while preserving the ciphertext size of $O(\log N)$. To achieve this, we introduce a new linear secret sharing scheme called TreeSSS, which requires a smaller number of shared keys and satisfies compactness. As a result, the Threshold Fully Homomorphic Encryption underlying our linear secret sharing scheme has fewer shared keys during the setup algorithm and reduced communication costs during the partial decryption algorithm. Moreover, the construction of a Universal Thresholdizer can be achieved through the use of TreeSSS, as it reduces the number of shared keys compared to previous constructions. Additionally, TreeSSS may be of independent interest, as it improves the efficiency in terms of communication costs when used to replace $\{0,1\}$-LSSS.

Keywords: Threshold Cryptography, Secret Sharing, Fully Homomorphic Encryption, Universal Thresholdizer

1 Introduction

A t-out-of-N threshold cryptography [19–21] is a public key cryptosystem that enables the distribution of secret keys among N parties. To obtain the plaintext from a ciphertext encrypted with this system, t parties are required to collaborate. However, even if $t - 1$ parties are compromised, they cannot gain any information about the plaintext.

The Universal Thresholdizer (UT) [9] is a tool for constructing threshold cryptosystems. It acts as a compiler, taking an existing cryptosystem and converting it into a threshold variant. UT provides simple constructions of threshold cryptographic primitives such as threshold signatures, CCA threshold PKE, and function secret sharing.

A black-box construction of UT was proposed that leverages compact threshold fully homomorphic encryption (TFHE) from learning with errors (LWE), non-interactive zero-knowledge proof with preprocessing (PZK) [30, 38], and a non-interactive commitment scheme [6]. This construction resolves a long-standing open question in lattice-based cryptography by constructing a one-round threshold signature using lattices.

There are two constructions of TFHE[4]: Shamir's secret sharing-based TFHE and $\{0,1\}$-linear secret sharing scheme (LSSS) based TFHE. The Shamir-based TFHE uses rational numbers, known as Lagrange coefficients, to distribute homomorphic encryption. However, this leads to a large scaling factor, resulting in a size of q that is not compact. Note that the size of scaling factor is $O(N!^2)$, so the bit-size of q is $O(N \log N)$.[5]

The TFHE based on the $\{0,1\}$-LSSS solves the limitation by utilizing a different approach to secret sharing. It adopts the monotone Boolean formula secret sharing scheme, as established by Valiant in [26,42], which employs binary coefficients to recover the secret from the distributed secret keys instead of the Lagrange coefficients used in Shamir's scheme. This allows compactness, with $\log q = O(\log N)$. However, this also leads to a high number of required secret keys, which can result in substantial communication overhead with a cost of $O(N^{4.3} \log N)$.

1.1 This work

Our main contribution is to propose a communication-efficient UT by constructing an efficient linear secret sharing scheme called TreeSSS. This new linear secret sharing scheme reduces the number of shared keys compared to the standard

[4] [9, Section 8.4] additionally introduces a type of TFHE that requires additional compilation steps. First, construct a non-compact TFHE from Shamir's secret sharing. Next, construct a non-compact UT from the non-compact TFHE. Then, construct a compact TFHE by taking as input another compact non-threshold FHE into the non-compact UT. Here, we will not consider this type of TFHE that requires further compilation.

[5] To put it simply, the property of compactness is maintained when the magnitude of q is bounded by a polynomial function of N.

Secret sharing scheme	$O(\log q)$	total # of keys	Comm cost
Shamir	$O(N \log N)$	N	$O(N \log N)$
$\{0,1\}$-LSSS	$O(\log N)$	$O(N^{5.3})$	$O(N^{4.3} \log N)$

Table 1: the state-of-the-art t-out-of-N TFHE.

$\{0,1\}$-LSSS, leading to less secret keys being shared in the setup algorithm of TFHE. This reduction in shared keys also reduces communication costs during the partial decryption algorithm of TFHE. Additionally, the compactness property of TFHE allows the primitive to be used in constructing UT.

For the purpose, we present a hybrid approach that integrates the $\{0,1\}$-LSSS and Shamir's Secret Sharing Scheme. We reinterpret a concrete implementation of the $\{0,1\}$-LSSS for threshold functions found in [29] as an iterative linear secret sharing method. The previous scheme utilized a well-known algorithm to convert a monotone Boolean circuit into a matrix based on the entire threshold circuit generated by monotone Boolean formulas. Instead of applying this algorithm to the entire threshold circuit, we observe that applying it to small matrices and iteratively multiplying these matrices can also generate the matrix corresponding to a threshold function. Furthermore, we replace the unit matrix used in the algorithm with the Vandermonde matrix used in Shamir's Secret Sharing, which leads to improve the efficiency in terms of communication costs.

This hybrid linear secret sharing scheme has the potential to decrease communication costs and may be of interest in various applications, such as threshold signatures [2], threshold multi-key FHE schemes [4], and decentralized ABE [43].

In conclusion, the hybrid approach to combining $\{0,1\}$-LSSS and Shamir's Secret Sharing Scheme results in improved efficiency, as evidenced in Table 2. Our findings align with the optimal formula for general majority functions built from $(2s-1)$-variable majority functions, as proven in [28] for $s \geq 2$. For further details, please refer to Section 2 and Section 4.

1.2 Related work

We briefly introduce related works of TFHE to present potential applications of our TFHE.

Comparison between Concurrent Work. Our approach to construct efficient TFHE is entirely distinct from that of the recent papers [10, 17, 31]. Our focus is on directly improving $\{0,1\}$-LSSS, while [10] uses it as is. We thus believe our TreeSSS and the technique in [10] can be combined to create an efficient TFHE and UT construction. We further note that [31] improved the bootstrapping technique of FHEW/TFHE to achieve threshold FHE, while [10, 17] both achieved a polynomial modulus-to-noise ratio TFHE from LWE using Rènyi divergence and the noise flooding technique. [17] was specialized for Torus-FHE

Secret Sharing Scheme		$O(\log q)$	total # of keys	Comm. cost
Previous	Shamir SS	$O(N \log N)$	N	$O(N \log N)$
	$\{0, 1\}$-LSSS	$O(\log N)$	$O(N^{5.3})$	$O(N^{4.3} \log N)$
Ours	TreeSSS with $\mathbf{V}_2$	$O(\log N)$	$O(N^{4.3})$	$O(N^{3.3} \log N)$
	TreeSSS with $\mathbf{V}_{10}$	$O(\log N)$	$O(N^{3.62})$	$O(N^{2.62} \log N)$
	TreeSSS with $\mathbf{V}_{50}$	$O(\log N)$	$O(N^{3.39})$	$O(N^{2.39} \log N)$
	TreeSSS with $\mathbf{V}_s$	$O(\log N)$	$O(N^{3+o(1)})$	$O(N^{2+o(1)} \log N)$

Table 2: The comparison results between the previous TFHE and ours. The size of ciphertexts is denoted by $O(\log q)$ and the number of keys required is represented by the total # of keys. The communication costs indicate the amount of information that each party must send for the partial decryption algorithm. $\mathbf{V}_s$ is the Vandermonde matrix which corresponds to s-out-of-$(2s-1)$ threshold structure. The last row shows the asymptotic behavior as N approaches infinity, with s held constant and N being sufficiently large.

[16], while [10] can be built from any FHE scheme. It is our expectation that integrating these techniques for optimizing the polynomial modulus-noise ratio with our TreeSSS approach would lead to an improvement in the TFHE construction.

Threshold Circuits. A majority circuit is equivalent to an $N/2$-out-of-N threshold circuit. Research has shown how to create monotone Boolean formulas for majority functions [26,28]. However, these constructions are not useful for threshold circuits similar to TFHE. For example, [26] showed that the depth of a three-variable majority function's monotone Boolean formula is 3, leading to a total number of secret shares close to $O(N^7)$, which is inefficient compared to circuit representations [42].

[28] found that the optimal formula for general majority can be expressed with the three-variable majority function in $O(N^{3+O(1/\log s)})$ for some s, which matches our main result.

Threshold Signature. The threshold signature is a protocol that uses the threshold property in a signature scheme. There have been many efforts to build this scheme [7, 8, 13, 18, 22–25, 33, 35, 41], but most of them are based on pre-quantum objects like ECDSA. There have only been two round-optimal threshold signature schemes from lattices [2, 9]. The first one was built from UT via compact TFHE [9], and the latter [2] improved efficiency through a concrete signature scheme and optimal noise flooding, providing a stronger security level using the random oracle model.

***N*-out-of-*N* TFHE.** N-out-of-N TFHE is a special case of TFHE and falls into the category of multikey FHE [3, 12, 27, 34, 36, 37]. It is a non-interactive protocol that allows for homomorphic computations on encrypted data using independently sampled keys, solving key management issues. It is considered a good solution for round-optimal secure multi-party computations [36] and on-the-fly MPC property [34].

1.3 Future Work

Our work has reduced the number of total share keys and communication cost in TFHE, but we believe that there still exists room for improvement. Our goal is to reduce the total number of keys to N, as in Shamir's secret sharing scheme, while maintaining $\log q = O(\log N)$.

2 Technical Overview

This section provides an overview of a new linear secret sharing scheme, called TreeSSS, and its application in threshold fully homomorphic encryption.

2.1 Previous Approach: $\{0,1\}$-LSSS for threshold functions from Monotone Boolean Formula

The $\{0,1\}$-LSSS is a family of linear secret sharing schemes that utilizes binary coefficients to recover the shared secret from partial secret shares, as defined in [9]. The use of monotone Boolean formulas [32] was proposed as an instantiation of $\{0,1\}$-LSSS. However, the polynomial-sized expression of threshold functions was proven by Valiant and Goldreich [26, 42]. Recently, [29] proposed using a folklore algorithm to demonstrate that monotone Boolean formulas are a part of $\{0,1\}$-LSSS. In this section, we briefly summarize the construction of threshold functions.

In this consideration, we focus on a threshold function with $N/2$-out-of-N parties, where N is even, for simplicity. Let φ be a level-0 formula which takes N bit-strings as input and returns one of the i-th input bits with some probability, where i is randomly chosen, or returns 0. For each $i \geq 1$, the level-$(i+1)$ formula is defined as $\varphi = (\varphi_1 \wedge \varphi_2) \vee (\varphi_3 \wedge \varphi_4)$, with $\varphi_1, \varphi_2, \varphi_3, \varphi_4$ randomly selected from a family of level-i formulas. Note that to maintain independence, the level-i formulas will not be duplicated.

In classic works [26, 42], it was proved that with $O(N^{5.3})$ level-0 formulas, a $N/2$-out-of-N threshold function can be expressed with a level-t formula with non-negligible probability, where $t = O(\log N)$. Building upon this result, [29] showed that this level-t formula can be converted into a $\{0,1\}$-LSSS for threshold functions.

To share a secret key $\mathsf{sk} \in \mathbb{Z}_q$, $\{0,1\}$-LSSS constructs a matrix $\mathbf{M} \in \mathbb{Z}_q^{\ell \times m}$, called the *share matrix*, with $m, \ell \gg N$, and distributes a subset of $\{w_i\}_{i \in [\ell]}$ to each party. The vector $\mathbf{w} = (w_i) = \mathbf{M} \cdot (\mathsf{sk}, r_2, \ldots, r_m)^T$ is computed using randomly sampled $r_i \leftarrow \mathbb{Z}_q$. The size of ℓ is equal to the size of level-t formula, $O(N^{5.3})$, and m is one more than the number of AND gates in level-t formula. This results in a total of $O(N^{5.3})$ secret shares. The $\{0,1\}$-LSSS for threshold functions in [29] is constructed as follows:

1. Consider level-0 formulas φ_i, where $i \in [O(N^{5.3})]$.
2. Create a level-(i+1) formula φ by combining $\varphi_1 \wedge \varphi_2$ and $\varphi_3 \wedge \varphi_4$ through an OR operation, where $\varphi_1, \varphi_2, \varphi_3, \varphi_4$ are randomly selected level-i formulas.
3. Repeat the process until i reaches t, which results in a level-t formula that is equivalent to the $N/2$-out-of-N threshold function with non-negligible probability.
4. Use the folklore algorithm to convert the level-t formula into a share matrix $\mathbf{M}$.

Note that throughout this paper, the folklore algorithm is considered a black-box method that converts circuits consisting of only AND and OR gates into matrices, except for this section. For more insightful discussion on the algorithm, please refer to [9, 29].

Input: A special monotone Boolean formula $C : \{0,1\}^n \to \{0,1\}$.
Output: A share matrix $\mathbf{M}$ induced by C.

1. Assign the label (1) to the root node of the tree, which represents a vector of length one.
2. Set a counter $\mathsf{count} = 1$. Then, proceed to label each node m in the tree, starting from the top and moving downwards.
3. For each non-leaf node m of the tree:
   (a) If m is an OR gate, assign the same label to its children as that of node m.
   (b) If m is an AND gate, first consider the labeled vector $\mathbf{v}$ on m. If required, pad $\mathbf{v}$ with 0's at the end to make it of the length count. Call the new vector $\mathbf{v}'$. Then label one child node with the vector $(\mathbf{v}', 1)$ and the other child node with the vector $(0, \ldots, 0, -1)$ which has length $\mathsf{count} + 1$.
4. Once all nodes in the tree are labeled, the vectors assigned to the leaf nodes form the rows of the matrix $\mathbf{M}$. If the lengths of these vectors differ, then pad the shorter ones with zeros at the end to make them have the same length.

Fig. 1: Folklore Algorithm in [29].

2.2 Attempt I: Regarding $\{0,1\}$-LSSS as Iterative Secret Sharing

As the first attempt, we reinterpret a secret sharing algorithm for threshold functions by utilizing the iterative steps of Boolean formula construction described in [42]. This allows us to construct the share matrix $\mathbf{M}$ through iterative matrix multiplications.

Observation. Section 2.1 describes the fact that the threshold circuit is an iterative construction of the Boolean monotone formulas: For i, the level-$(i+1)$ formula $\varphi^{(i+1)}$ is generated from four level-i formulas, $\varphi_1^{(i)}, \varphi_2^{(i)}, \varphi_3^{(i)}$ and $\varphi_4^{(i)}$. Specifically, $\varphi^{(i+1)} = (\varphi_1^{(i)} \wedge \varphi_2^{(i)}) \vee (\varphi_3^{(i)} \wedge \varphi_4^{(i)})$.

We first observe that the relation between $\varphi^{(i+1)}$ and $\{\varphi_j^{(i)}\}j \in \{1, 2, 3, 4\}$ can be represented as a binary tree of depth 2, as in the structure shown in Figure 2. This tree is composed of AND and OR gates, allowing us to utilize the folklore algorithm in the above section. There exists a small matrix $\mathbf{D}$ that corresponds to this binary tree, with the leaf nodes being $\{\varphi_j^{(i)}\}j \in \{1, 2, 3, 4\}$. Here, $\mathbf{D}$ is a 4×2 matrix defined as

$$\mathbf{D} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & -1 \\ 0 & -1 \end{bmatrix} \in \mathbb{Z}_q^{4\times 2}.$$

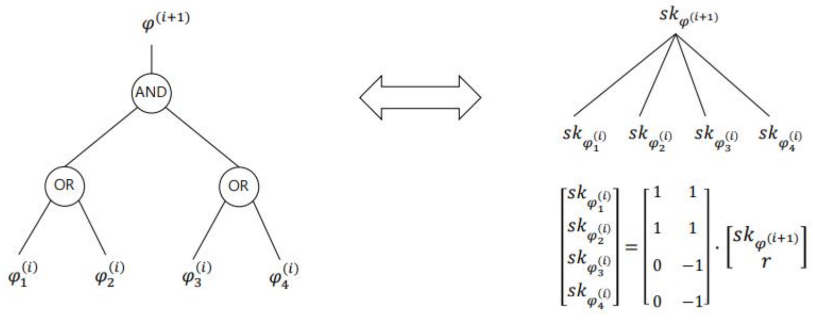
$\varphi^{(i+1)}$
AND
OR
OR
$\varphi_1^{(i)}$
$\varphi_2^{(i)}$
$\varphi_3^{(i)}$
$\varphi_4^{(i)}$
$sk_{\varphi^{(i+1)}}$
$sk_{\varphi_1^{(i)}}$
$sk_{\varphi_2^{(i)}}$
$sk_{\varphi_3^{(i)}}$
$sk_{\varphi_4^{(i)}}$
$\begin{bmatrix} sk_{\varphi_1^{(i)}} \\ sk_{\varphi_2^{(i)}} \\ sk_{\varphi_3^{(i)}} \\ sk_{\varphi_4^{(i)}} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & -1 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} sk_{\varphi^{(i+1)}} \\ r \end{bmatrix}$

Fig. 2: Boolean formula corresponds to secret share

Furthermore, the correspondence between the binary tree and the matrix is established through the relationship

$$\begin{bmatrix} \mathsf{sk}_{\varphi_1^{(i)}} \\ \mathsf{sk}_{\varphi_2^{(i)}} \\ \mathsf{sk}_{\varphi_3^{(i)}} \\ \mathsf{sk}_{\varphi_4^{(i)}} \end{bmatrix} = \mathbf{D} \cdot \begin{bmatrix} \mathsf{sk}_{\varphi^{(i+1)}} \\ r \end{bmatrix}$$

where $r \in \mathbb{Z}_q$ is a random integer. Thus, the operation $\varphi^{(i+1)} = (\varphi_1^{(i)} \wedge \varphi_2^{(i)}) \vee (\varphi_3^{(i)} \wedge \varphi_4^{(i)})$ can be viewed as a matrix multiplication with $\mathbf{D}$. Similarly, the representation of the formula $\varphi^{(i+1)}$ from 16 $\varphi^{(i-1)}$ formulas can be represented as a matrix $\mathbf{I}_4 \otimes \mathbf{D} \in \mathbb{Z}^{16\times 8}$, where $\mathbf{I}_4$ is the 4-dimensional identity matrix and $\mathbf{A} \otimes \mathbf{B}$ is the Kronecker tensor product of matrices $\mathbf{A}$ and $\mathbf{B}$. Consequently, there is a matrix $\mathbf{M}$ which corresponds to circuit representations of level-t formula $\varphi^{(t)}$ from level-0 $\varphi^{(0)}$ formulas.

As a result, the matrix-vector product $\mathbf{w} = \mathbf{M} \cdot \mathbf{v}$ gives us the sharing of the secret. Here, $\mathbf{v} = (\mathsf{sk}_{\varphi^{(t)}}, r_2, \ldots, r_m)^T$ and w_i is the i-th coordinate of $\mathbf{w}$ which is distributed to each party. The result is given by Proposition 2.1 and we defer the proof in the Supplementary material A.1.

Proposition 2.1 *Given a linear secret sharing scheme* $\mathsf{SS}^{(1)}$ *with a share matrix* $\mathbf{M}^{(L)} \in \mathbb{Z}_q^{\ell \times m}$ *and an iteration number* L*, the* L*-th iterative construction* $\mathsf{SS}^{(L)}$ *also provides a linear secret sharing scheme.*

It should be noted that this iterative construction cannot reduce the size of $\mathbf{M}$, as it operates as a mapping between binary trees and small matrices. The lesson from this observation is

- It is not necessary to apply the folklore algorithm to the entire threshold circuit in order to obtain the $\{0,1\}$-LSSS.
- The share matrix $\mathbf{M}$ of $\{0,1\}$-LSSS can be constructed through simple matrix multiplications of small size.

2.3 Attempt II: TreeSSS - change the underlying matrix into the Vandermonde matrix

Valiant's monotonic Boolean construction for majority functions consists exclusively of AND and OR gates, which can be easily transformed into matrices through the application of the folklore algorithm. In contrast, alternative constructions for N-input majority functions have been proposed by Gupta and Mahajan [28] and Goldreich [26], which are based on $(2s-1)$-input majority functions with $s \ll N$. These alternative constructions depart from the traditional AND and OR gates by replacing them with $(2s-1)$-input majority functions. Our approach mirrors this shift, as we replace the matrix $\mathbf{D}$ with the Vandermonde matrix to provide a matrix-based construction for majority functions.

To demonstrate this concept, we present the following lemma, which outlines the construction of majority functions from a 3-input majority function. The lemma asserts that the utilization of small input majority functions can increase the probability.

Lemma 2.2 ([26, 28]) *Let* $X_1, X_2, X_3 \leftarrow \{0,1\}$ *be three independent identically distributed random variables, and* $p := \Pr[X_i = 1]$ *for all* i*. Then, the following holds:*

1. $p' := \Pr[\mathit{MAJ}_3(X_1, X_2, X_3) = 1] = p^3 + 3 \cdot (1-p) \cdot p^2$.
2. $\delta := p - 0.5$*, it holds that* $p' = 0.5 + (1.5 - 2\delta^2) \cdot \delta$.
3. $p' < 3p^2$.

The lemma posits that the generation of the N-input majority function can be accomplished through the application of the 3-input majority function in a tree structure, thus reducing both the level of approximation and the size of the formula. A proof of this lemma can be found in the supplementary material A.2. The findings of [28] indicate that the construction of majority functions with N inputs is possible through the use of a trivariate majority function with a size of $O(N^{4.29})$ or through a $(2s-1)$-input majority function with a size of $O(N^{3+O(1/\log s)})$.

However, the folklore algorithm that converts circuits into matrices cannot be applied to the constructions proposed by [28] and [26]. This is because the folklore algorithm is limited to circuits that consist only of AND and OR gates, while the cost of converting the circuit using this algorithm becomes prohibitively high when compared to converting Valiant's circuit [26].

As a result, the findings of [26, 28] cannot be immediately utilized to create an efficient linear secret sharing scheme for threshold functions. Our proposed TreeSSS algorithm for threshold functions, presented in Section 2.2, represents the start of matrix-based linear secret sharing. The central idea is to use a different matrix $\mathbf{V}$ that corresponds to small input majority functions, which has the potential to bypass the folklore algorithm in the creation of an efficient scheme.

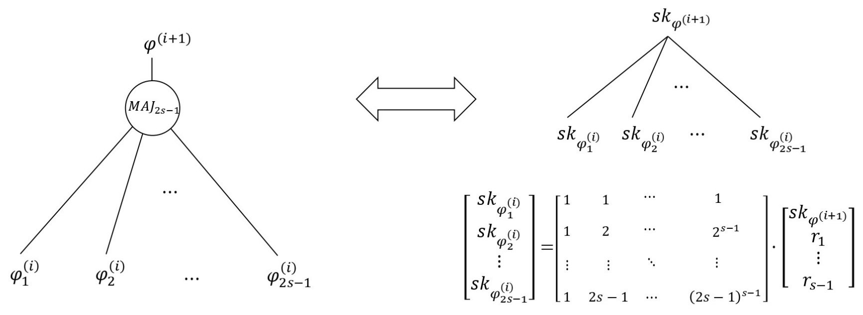
$\varphi^{(i+1)}$
MAJ_{2s-1}
$\varphi_1^{(i)}$
$\varphi_2^{(i)}$
$\cdots$
$\varphi_{2s-1}^{(i)}$
$sk_{\varphi^{(i+1)}}$
$sk_{\varphi_1^{(i)}}$
$sk_{\varphi_2^{(i)}}$
$sk_{\varphi_{2s-1}^{(i)}}$
$$\begin{bmatrix} sk_{\varphi_1^{(i)}} \\ sk_{\varphi_2^{(i)}} \\ \vdots \\ sk_{\varphi_{2s-1}^{(i)}} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & 2^{s-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2s-1 & \cdots & (2s-1)^{s-1} \end{bmatrix} \cdot \begin{bmatrix} sk_{\varphi^{(i+1)}} \\ r_1 \\ \vdots \\ r_{s-1} \end{bmatrix}$$

Fig. 3: Small input-Majority function corresponds to Shamir's secret sharing

The utilization of the Vandermonde matrix $\mathbf{V}_s \in \mathbb{Z}_q^{(2s-1)\times s}$ in our approach offers a hybrid strategy that combines the features of Shamir's secret sharing and the iterative construction. The choice of this matrix is motivated by its proven performance in the context of Shamir's secret sharing, as demonstrated in [39]. For a visual representation of this concept, refer to Figure 3. The results of our construction match the previously established results by [28]. The detailed construction and findings are presented in Section 4.2.

We further remark that the simple replacement of the matrix has an unexpected impact on the simulation security proof of TFHE. We recall the concept of the partial decryption algorithm. The algorithm works by taking the secret shares sk_i and computing $\mathsf{sk} = \sum_i c_i \cdot \mathsf{sk}_i$, where c_i are the recovery coefficients and sk is the master secret key. The decryption process is performed as follows:

$$\langle \mathsf{ct}, \mathsf{sk} \rangle = \langle \mathsf{ct}, \sum_i c_i \cdot \mathsf{sk}_i \rangle = \sum_i c_i \langle \mathsf{ct}, \mathsf{sk}_i \rangle \bmod q$$

Consequently, the partial decryption algorithm can be regarded as follows[6]:

$$\langle \mathsf{ct}, \mathsf{sk}_i \rangle = (c_i^{-1} \bmod q) \cdot \left(\langle \mathsf{ct}, \mathsf{sk} \rangle - \sum_{i \neq j} \langle \mathsf{ct}, \mathsf{sk}_j \rangle \bmod q \right) \bmod q.$$

The observation is critical in the simulation security proof as it enables the simulator to simulate the partial decryption algorithm without having any knowledge of the secret shares sk_i. This issue is not relevant in the case of $\{0,1\}$-LSSS, as the recovery coefficients c_i and their inverse elements c_i^{-1} are binary. However, in the hybrid approach, the coefficients are Lagrange's coefficients, which implies that there is no guarantee of the smallness of the inverse elements of Lagrange's coefficients.

Therefore, to adapt the simulation security proof for TFHE, it is necessary to provide an upper bound on the inverse of the Lagrange's coefficients. This enables us to overcome the smallness issue, which is a major concern in the simulation security proof of linear secret sharing schemes. It is worth mentioning that in [29], the authors proposed the $\{0,1\}$-LSSS to avoid the smallness issue, which is one of the ways to overcome this challenge in the simulation security proof.

To conclude, we revisit and slightly modify the previous results of the Lagrange coefficients presented in [1,40]. The result is a new lemma which completes the security proof for the proposed construction. The proof of the lemma can be found in the Supplementary material A.3.

Lemma 2.3 *Let $P = P_1, \cdots, P_N$ be a set of parties and $\mathbb{A}_t$ a threshold access structure on P with a threshold value $t \in [N]$. Consider a Shamir's secret sharing scheme SS over the secret space $\mathbb{Z}_q$, where q is a prime number such that $(N!)^2 \leq q$. Then, for any set $S \subseteq [N] \cup 0$ with size t and for any indices $i, j \in [N]$, the following properties hold:*

- $|N! \cdot \lambda_{i,j}^S| \leq (N!)^2$, $\left| N! \cdot \frac{1}{\lambda_{i,j}^S} \right| \leq (N!)^2$,
- $N! \cdot \lambda_{i,j}^S, N! \cdot \frac{1}{\lambda_{i,j}^S}$ *are integers*

where $\lambda_{i,j}^S$ is the Lagrange coefficient.

TreeSSS for t-out-of-N structures. We can easily extend our construction to t-out-of-N TreeSSS by slightly modifying the number of parties and the threshold. The conversion process involves constructing a TreeSSS for a different t and N that satisfies the desired conditions. For instance, when $t > \lceil \frac{N}{2} \rceil$, we start by

[6] To prevent information leakage, the large error should be added. However, we omit the error for simplicity.

constructing a TreeSSS for t-out-of-$(2t-1)$. If $2t-1 > N$, we simply disregard the extra secret shares. Similarly, if $t < \lceil \frac{N}{2} \rceil$, we generate a TreeSSS for $(t+r)$-out-of-$(N+r)$ where r satisfies $\frac{N+r+1}{2} = t+r$. In the case where $t = \frac{N}{2}$, and N is even, it suffices to construct a TreeSSS for $(\frac{N}{2}+1)$-out-of-$(N+1)$. The detailed construction can be found in Section 4.3.

3 Preliminary

Notations. We use bold uppercase letters for matrices and bold lowercase letters for vectors. The set $[n] = 1, 2, \cdots, n$ is used to denote a positive integer n. log is used to represent the logarithm function base 2. The size of a finite set S is represented by $|S|$ and its power set is represented by $\mathcal{P}(S)$. $a \leftarrow S$ means that a is randomly selected from the finite set S. $D_1 \approx_s D_2$ means that the distribution D_1 is statistically indistinguishable from distribution D_2.

Vandermonde Matrix. We use the Vandermonde matrix, a special matrix widely used in Shamir's secret sharing scheme, and denote it as $\mathbf{V}_{N,t}$, where it is a $N \times t$ matrix. The entries in $\mathbf{V}_{N,t}$ are defined as:

$$\mathbf{V}_{N,t} = \begin{bmatrix} 1 & 1 & 1^2 & \cdots & 1^{t-1} \\ 1 & 2 & 2^2 & \cdots & 2^{t-1} \\ 1 & 3 & 3^2 & \cdots & 3^{t-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & N & N^2 & \cdots & N^{t-1} \end{bmatrix}.$$

For convenience, we also use the shorthand notation $\mathbf{V}_s$ to refer to $\mathbf{V}_{2s-1,s}$.

Kronecker Tensor Product. Given two matrices $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and $\mathbf{B} \in \mathbb{Z}_q^{r \times s}$, the Kronecker tensor product, denoted as $\mathbf{A} \otimes \mathbf{B}$, is defined as:

$$\begin{bmatrix} a_{1,1} \cdot \mathbf{B} & a_{1,2} \cdot \mathbf{B} & \cdots & a_{1,n} \cdot \mathbf{B} \\ a_{2,1} \cdot \mathbf{B} & a_{2,2} \cdot \mathbf{B} & \cdots & a_{2,n} \cdot \mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} \cdot \mathbf{B} & a_{m,2} \cdot \mathbf{B} & \cdots & a_{m,n} \cdot \mathbf{B} \end{bmatrix} \in \mathbb{Z}_q^{mr \times ns}$$

Statistical Distance. he distance between two distributions D_1 and D_2 over a countable support X is defined as

$$\Delta(D_1, D_2) = \frac{1}{2} \sum_{e \in E} \left| \Pr_{e \leftarrow D_1}(D_1(e)) - \Pr_{e \leftarrow D_2}(D_2(e)) \right|.$$

The noise flooding technique, also known as noise smudging, is commonly used to mask information by adding a large error.

Lemma 3.1 (Noise Flooding Technique [3, 36]) *Let B_1, B_2 be positive integers and e_1 be an integer in the interval $[-B_1, B_1]$. Let U be a uniform distribution over the interval $[-B_2, B_2]$. Then, it holds that $\Delta(U, U + e_1) \leq \frac{B_1}{B_2}$.*

Learning with Errors (LWE). The Learning with Errors (LWE) problem is a fundamental problem in lattice-based cryptography, often used in the construction of fully homomorphic encryption schemes [11, 14, 16].

Given positive integers n, m, and q and a noise distribution χ over $\mathbb{Z}_q$, the $\mathsf{LWE}(n, m, q, \chi)$ problem involves an adversary attempting to distinguish between two distributions: $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$ and $(\mathbf{A}, \mathbf{u})$. Here, $\mathbf{A}$ is chosen from $\mathbb{Z}_q^{m \times n}$, $\mathbf{s}$ is chosen from $\mathbb{Z}_q^n$, $\mathbf{e}$ is chosen from χ^m, and $\mathbf{u}$ is randomly chosen from $\mathbb{Z}_q^m$.

3.1 Majority Circuits

we briefly introduce definitions and previous results for majority functions which are equivalent to threshold functions.

Definition 3.2 (Monotone Boolean formula [9]) *A Boolean circuit $C : \{0,1\}^N \to \{0,1\}$ is called a monotone Boolean formula if it satisfies the following conditions:*

- *It has a single output gate.*
- *Each gate is either an AND or an OR gate with a fan-in of 2 and a fan-out of 1.*
- *The input wires may have multiple connections to other gates*

Definition 3.3 (Majority Function/Gate) *A majority function/gate $\mathsf{MAJ}_N : \{0,1\}^N \to \{0,1\}$ is a function defined as follows:*

$$\mathsf{MAJ}_N(x) = \begin{cases} 1 & wt(x) \geq N/2 \\ 0 & otherwise, \end{cases}$$

where $wt(x)$ is the number of nonzero bits in $x = x_1 x_2 \dots x_N$

We now provide a summary of results that demonstrate the construction of majority functions from monotone Boolean formulas, as proven by [26, 28, 42]. However, please refer to the original papers for a full understanding and proof of these results.

Lemma 3.4 ([42]) *Let $F : \{0,1\}^N \to \{0,1\}$ be a randomized function, and $x \in \{0,1\}^N$ be its input. The probability of $F(x)$ is as follows:*

$$\Pr[F(x) = 1 \mid wt(x) < N/2] < 2^{-N-1}$$
$$\Pr[F(x) = 0 \mid wt(x) \geq N/2] < 2^{-N-1}.$$

Then, it satisfies that $\Pr[F \equiv \mathsf{MAJ}_N] \geq 1/2$.

Using Lemma 3.4, Valiant proposed a (recursive) construction of monotone Boolean formula for computing the majority function. The unit circuit $\varphi^{(i+1)} = (\varphi_1^{(i)} \wedge \varphi_2^{(i)}) \vee (\varphi_3^{(i)} \wedge \varphi^{(i)})_4$ is repeated to construct the majority function for N. [42] showed that repeating the unit circuit up to level-$2.65 \log N$ can construct the majority function for N.

Lemma 3.5 ([42]) *Let $\gamma = 2(3 - \sqrt{5}) \approx 1.52$, and N be an even number of parties. If the unit circuit $\varphi = (\varphi_1 \wedge \varphi_2) \vee (\varphi_3 \wedge \varphi_4)$ is iteratively constructed with an iteration number $L \geq \log_\gamma N + \log N + O(1)$, then there exists an $O(N^{5.3})$-size monotone formula for computing the majority function MAJ_N.*

Note that the construction presented in Lemma 3.6 involves converting the MAJ_3 gate into a formula consisting only of AND/OR gates, which results in a larger formula compared to the construction in [42]. The conversion of the MAJ_3 gate into AND/OR gates involves replacing $\mathsf{MAJ}_3(F_1, F_2, F_3)$ with $(F_1 \wedge F_2) \vee (F_2 \wedge F_3) \vee (F_3 \wedge F_1)$. However, this conversion is necessary as there is currently no known way to convert majority gates in circuits into matrices, unlike AND/OR gates, which can be converted via the folklore algorithm.

Lemma 3.6 ([26]) *Let N be the number of parties with odd value. Then, there exists a construction of the majority function MAJ_N using the small majority gate MAJ_3. Specifically, MAJ_N can be constructed from MAJ_3 with a total of $L \geq \log_{1.5}(N) + \log_2 N + O(1)$ iterations. As a result, there exists a monotone Boolean formula for computing MAJ_N of size $O(N^7)$.*

In [28], the authors studied the construction of majority functions from small majority gates. They leveraged the technique of amplifying the probability of majority gate applications. This lemma states that given an odd number of parties N, a construction of the majority function MAJ_N from MAJ_{2s-1}-gates is possible.

Lemma 3.7 ([28]) *Given an odd number N of parties, there exists a construction of the majority function MAJ_N from small majority gates MAJ_{2s-1}. This construction requires iteratively applying MAJ_{2s-1}-gates up to level $\log_{c_s}(N) + \log_s N + O(1)$, where $c_s = \frac{2s-1}{2^{2s-2}} \cdot \binom{2s-2}{s-1}$.*

In our analysis, we will demonstrate a new secret sharing approach that utilizes Shamir's secret sharing and the above lemma, bypassing the need for the folklore lemma. This will be presented in section 4

3.2 Fully Homomorphic Encryption

We recall the definition of fully homomorphic encryption and its properties.

Definition 3.8 (Fully homomorphic encryption) *An FHE scheme is described by a set of algorithms with the following properties:*

- *The setup algorithm* $\mathsf{FHE.Setup}(1^\lambda, 1^d)$ *takes as input the security parameter* λ*, and a depth bound* d*, and outputs a pair of the public key* pk *and secret key* sk*.*
- *The encryption algorithm* $\mathsf{FHE.Enc}(\mathsf{pk}, \mu)$ *takes as input* pk *and a message* $\mu \in \{0,1\}$*, and outputs a ciphertext* ct*.*
- *The evaluation algorithm* $\mathsf{FHE.Eval}(C, \mathsf{ct}_1, \cdots, \mathsf{ct}_l, \mathsf{pk})$ *takes as input* l*-input circuit* C *with less than or equal depth* d*, a bunch of ciphertexts* $\mathsf{ct}_1, \cdots, \mathsf{ct}_l$ *and* pk*, and outputs an evaluated ciphertext* $\hat{\mathsf{ct}}$*.*
- *The decryption algorithm* $\mathsf{FHE.Dec}(\mathsf{pk}, \mathsf{sk}, \hat{\mathsf{ct}})$ *takes as input* pk*,* sk *and* $\hat{\mathsf{ct}}$*, and outputs a message* $\mu \in \{0,1\}$*.*

Hereafter, we use notations from Definition 3.8.

Definition 3.9 (Evaluation Correctness) *We say* FHE *scheme is correct if for any evaluated ciphertext* $\hat{\mathsf{ct}}$ *generated by* $\mathsf{FHE.Eval}(C, \mathsf{ct}_1, \cdots, \mathsf{ct}_l, \mathsf{pk})$ *satisfies*

$$\Pr[\mathsf{FHE.Dec}(\mathsf{pk}, \mathsf{sk}, \hat{\mathsf{ct}}) = C(\mu_1, \cdots, \mu_l)] = 1 - \mathsf{negl}(\lambda)$$

Definition 3.10 (Compactness) *We say* FHE *scheme is compact if for any ciphertext* ct *generated from the algorithm of* $\mathsf{FHE.Enc}$*, there is a polynomial* poly *such that* $|\mathsf{ct}| \le \mathsf{poly}(\lambda, d)$*.*

Definition 3.11 (Semantic security) *We say that* FHE *is secure if for all security parameter* λ *and depth bound* d*, the following holds: for any PPT adversary* $\mathcal{A}$*, the experiment* $Expt_{\mathcal{A},\mathsf{FHE}}(1^\lambda, 1^d)$ *outputs 1 except for negligible probability:*

$Expt_{\mathcal{A},\mathsf{FHE}}(1^\lambda, 1^d)$:

1. *Given* (λ, d)*, the challenger runs* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$ *and* $\mathsf{ct} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, b)$ *for* $b \leftarrow \{0,1\}$*.*
2. *The challenger sends* $(\mathsf{pk}, \mathsf{ct})$ *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs a guess* b'*.*
4. *The experiment outputs 1 if* $b = b'$*.*

Definition 3.12 (Special FHE) *We say* FHE *is a special* FHE *if it satisfies the following properties:*

- *The setup algorithm* $\mathsf{Setup}(1^\lambda, 1^d)$ *takes as input the security parameter* λ *and a depth bound* d*, and outputs* $(\mathsf{pk}, \mathsf{sk})$*, where* pk *contains a prime* q*, and* $\mathsf{sk} \in \mathbb{Z}_q^n$ *for some* $n = \mathsf{poly}(\lambda, d)$*.*
- *The decryption algorithm* Dec *consists of two functions* $(\mathsf{Dec}_0, \mathsf{Dec}_1)$ *defined as follows:*
  - $p \leftarrow \mathsf{Dec}_0(\mathsf{sk}, \mathsf{ct})$*:* p *is of the form* $\mu \cdot \lfloor \frac{q}{2} \rceil + e$ *for a noise* $e \in [-cB, cB]$ *with the noise bound* $B = B(\lambda, d, q)$*. Here,* e *is an integer multiple of* c*.*

- $\mu \leftarrow \mathsf{Dec}_1(p)$: *Given p, return* $\mu = \begin{cases} 0 & \text{if } p \in [-\lfloor \frac{q}{4} \rceil, \lfloor \frac{q}{4} \rceil] \\ 1 & \text{otherwise.} \end{cases}$

- Dec_0 *is a linear function over $\mathbb{Z}_q$ such as inner product and matrix multiplication in the secret key sk.*

The constant c in Definition 3.12 is called the *multiplicative constant.*

3.3 Non-interactive Zero Knowledge Proof and Commitments

This section introduces the building blocks for constructing the universal thresholdizer, which are not defined in the main body of this paper. The descriptions of these schemes are based on [9].

These building blocks have been utilized in the construction of a universal thresholdizer in a black-box manner.

Definition 3.13 (Non-interactive Zero Knowledge Proof with pre-processing) *A non-iterative zero-knowledge proof with pre-processing (PZK) for a language L with a relation R is a tuple of PPT algorithms $\mathsf{PZK} = (\mathsf{PZK.Pre}, \mathsf{PZK.Prove}, \mathsf{PZK.Verify})$. The output of the pre-processing algorithm $\mathsf{PZK.Pre}(1^\lambda)$ is a pair of systems (σ_P, σ_V). The PZK scheme must satisfy the following properties:*

- ***Completeness**: For every $(x, w) \in R$, the probability that the verifier will accept a proof generated by the prover is 1, i.e.*

$$\Pr[\mathsf{PZK.Verify}(\sigma_V, x, \pi) = 1 : \ \pi \leftarrow \mathsf{PZK.Prove}(\sigma_P, x, w)] = 1$$

- ***Soundness**: For every $x \notin L$, the probability of the existence of a proof $\pi \leftarrow \mathsf{PZK.Prove}(\sigma_P, x, w)$ such that $\Pr[\mathsf{PZK.Verify}(\sigma_V, x, \pi) = 1]$ is negligible in λ.*
- ***Zero knowledge**: There is a PPT simulator S such that for any $(x, w) \in R$, no one can computationally distinguish two distributions:*

$$\{\sigma_V, \mathsf{PZK.Prove}(\sigma_P, x, w)\} \approx \{S(x)\}$$

Lemma 3.14 ([30, 38]) *PZK can be constructed from one-way functions.*

We now introduce a new component for the universal thresholdizer, as described in [5, 9].

Definition 3.15 (Non-interactive Commitment [6]) *We say that $\mathsf{C} = (\mathsf{C.Com})$ is a non-interactive commitment scheme if the following holds: Let com be a string in $\{0,1\}^*$ outputted by $\mathsf{C.Com}(\mathbf{x}; \mathbf{r})$ for a message $\mathbf{x} \in \{0,1\}^*$ with randomness $\mathbf{r} \in \{0,1\}^\lambda$. Then,*

- ***Perfect binding**: For any security parameter $\lambda \in \mathbb{N}$, and randomness $\mathbf{r}_0, \mathbf{r}_1 \in \{0,1\}^\lambda$, if $\mathsf{C.Com}(\mathbf{x}_0; \mathbf{r}_0) = \mathsf{C.Com}(\mathbf{x}_1; \mathbf{r}_1)$, then it holds that $\mathbf{x}_0 = \mathbf{x}_1$.*

- ***Computational hiding**: For any security parameter $\lambda \in \mathbb{N}$ and $\mathbf{x}_0, \mathbf{x}_1 \in \{0,1\}^{poly(\lambda)}$, no PPT adversary can distinguish the following distributions:*

$$\{\mathsf{com}_0 : \mathbf{r} \leftarrow \{0,1\}^\lambda, \mathsf{com}_0 \leftarrow \mathsf{C.Com}(\mathbf{x}_0; \mathbf{r})\}$$
$$\approx \{\mathsf{com}_1 : \mathbf{r} \leftarrow \{0,1\}^\lambda, \mathsf{com}_1 \leftarrow \mathsf{C.Com}(\mathbf{x}_1; \mathbf{r})\}$$

Lemma 3.16 ([6]) *A non-interactive commitment can be built from injective one-way functions.*

4 The Tree Secret Sharing algorithm for majority function

This section presents a key technical contribution of this paper, the tree secret sharing scheme (TreeSSS).

4.1 Preliminaries for Secret Sharing

This section provides several relevant definitions for secret sharing schemes and Shamir's secret sharing scheme as a representative. For this purpose, we adopt definitions/notations from [9].

Definition 4.1 (Threshold Structure) *Given a set of parties $P = P_1, \ldots, P_N$ and a threshold value t such that $1 \leq t \leq N$, the t-out-of-N threshold structure $\mathbb{A}_t \subseteq \mathcal{P}(P)$ is defined as the collection of all subsets $S \in \mathcal{P}(P)$ with a size of at least t. The subsets in $\mathbb{A}_t$ are referred to as "valid sets," and the subsets in $\mathcal{P}(P) \setminus \mathbb{A}_t$ are referred to as "invalid sets."*

Now, we define the secret sharing scheme for the threshold structure.

Definition 4.2 (Secret Sharing Scheme) *Let $\mathcal{K}$ be the secret key space. The secret sharing scheme SS is defined as a pair of PPT algorithms, $(\mathsf{SS.Share}, \mathsf{SS.Combine})$:*

- *The share algorithm $\mathsf{SS.Share}(\mathsf{sk}, \mathbb{A}_t)$ takes input a secret key $\mathsf{sk} \in \mathcal{K}$ and threshold structure $\mathbb{A}_t$ on P and returns a family of secret shares $\mathsf{sk}_1, \cdots, \mathsf{sk}_N$ for each party.*
- *The combine algorithm $\mathsf{SS.Combine}(B)$ takes input a set of shares $B = \bigcup_{P_i \in S}\{\mathsf{sk}_i\}$ and returns $\hat{\mathsf{sk}}$.*

A secret sharing scheme must satisfy the following properties.

Definition 4.3 (Correctness) *For every $S \in \mathbb{A}_t$, $\mathsf{sk} \in \mathcal{K}$, and a set of shares $\{\mathsf{sk}_i\}_{i \in [N]}$ obtained by the share algorithm which takes as input sk and $\mathbb{A}_t$, the following holds without negligible probability:*

$$\mathsf{SS.Combine}(\{\mathsf{sk}_i\}_{i \in S}) = \begin{cases} \mathsf{sk} & \text{for } S \in \mathbb{A}_t \\ \perp & \text{for } S \notin \mathbb{A}_t \end{cases}$$

Definition 4.4 (Privacy) *For all $S \notin \mathbb{A}$ and $\mathsf{sk}_0, \mathsf{sk}_1 \in \mathcal{K}$, two sets of shares $(\mathsf{sk}_{b,1}, \cdots, \mathsf{sk}_{b,N}) \leftarrow \mathsf{SS.Share}(\mathsf{sk}_b, \mathbb{A}_t)$ for $b \in \{0,1\}$ follow the identical distribution*

$$\{\mathsf{sk}_{0,i}\}_{i\in S} \approx \{\mathsf{sk}_{1,i}\}_{i\in S}.$$

Especially, we introduce a linear secret sharing scheme (or linear threshold secret sharing scheme) to construct the threshold FHE.

Definition 4.5 (Linear Secret Sharing Scheme (LSSS)) *A secret sharing scheme SS with secret space $K = \mathbb{Z}_q$ is called a linear secret sharing scheme if the following properties are satisfied:*

- *$\mathsf{SS.Share}(\mathsf{sk}, \mathbb{A}_t)$: There exists a share matrix $\mathbf{M} \in \mathbb{Z}_q^{\ell\times n}$ with positive integers ℓ, m and associate a partition T_i of $[\ell]$ to each party P_i. For a given secret $\mathsf{sk} \in \mathbb{Z}_q$, the sharing algorithm samples random values $r_2, \cdots, r_n \leftarrow \mathbb{Z}_q$ and generates a vector $(\mathsf{share}_1, \cdots, \mathsf{share}_\ell)^T = \mathbf{M}\cdot(\mathsf{sk}, r_2, \cdots, r_n)^T$. The share for P_i is a set of entries $\mathsf{sk}_i = \{\mathsf{share}_j\}_{j\in T_i}$.*

- *$\mathsf{SS.Combine}(B)$: For any $S \in \mathbb{A}_t$, one can efficiently find the coefficient $\{c_j^S\}_{j\in\bigcup_{P_i\in S} T_i}$ such that*

$$\sum_{j\in\bigcup_{P_i\in S} T_i} c_j^S \cdot M[j] = (1, 0, \cdots, 0).$$

  Then,S can recover a secret key sk by computing $\mathsf{sk} = \sum_{j\in\bigcup_{P_i\in S} T_i} c_j^S \cdot \mathsf{share}_j$. The coefficients $\{c_j^S\}$ are called recovery coefficients.

Shamir's secret sharing scheme. Shamir's secret sharing scheme is a linear secret sharing scheme whose share matrix is a Vandermonde matrix.

Definition 4.6 (Shamir's secret sharing [39]) *Let $P = \{P_1, \cdots, P_N\}$ be a set of parties and let $\mathbb{A}_t$ be the t-out-of-N threshold structure on P. The Shamir's secret sharing scheme SS for a secret key space $\mathcal{K} = \mathbb{Z}_q$ is a tuple of PPT algorithms $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Combine})$ defined as follows:*

- *$\mathsf{SS.Share}(\mathsf{sk}, \mathbb{A}_t) \to (\mathsf{sk}_1, \cdots, \mathsf{sk}_N)$: There exists a vandermonde matrix $\mathbf{V}_{N,t}$ for t-out-of-N threshold structure. On input a secret $\mathsf{sk} \in \mathcal{K}$ and threshold structure $\mathbb{A}_t$ on P, the sharing algorithm samples random values $r_2, \cdots, r_t \leftarrow \mathbb{Z}_q$ and computes a secret share vector $(\mathsf{sk}_1, \cdots, \mathsf{sk}_N)^T = \mathbf{V}_{N,t}\cdot(\mathsf{sk}, r_2, \cdots, r_t)^T$ and distributes a set of shares $\mathsf{sk}_1, \cdots, \mathsf{sk}_N$ for each party.*
- *$\mathsf{SS.Combine}(B) \to \hat{\mathsf{sk}}$: On input a set of shares $B = \bigcup_{P_i\in S}\{\mathsf{sk}_i\}$, the Lagrange coefficients obtained from the Lagrange polynomial satisfies following equality:*

$$\lambda_i^S := \prod_{P_j\in S\setminus\{P_i\}} \frac{-j}{i-j}, \sum_{P_i\in S} \lambda_i^S \cdot \mathbf{V}_{N,t}[i] = (1, 0, \cdots, 0).$$

Then, the combining algorithm outputs $\hat{\mathsf{sk}} = \sum_{P_i \in S} \lambda_i^S \cdot \mathsf{sk}_i$ *which is equal to* sk.

Theorem 4.7 **([39])** *Let* $P = \{P_1, \cdots, P_N\}$ *be a set of parties and let* $\mathbb{A}_t$ *be the* t*-out-of-*N *threshold structure on* P*. Then, Shamir's secret sharing* SS *with secret space* $\mathcal{K} = \mathbb{Z}_q$ *for some prime* q *satisfies the following properties:*

1. *For any* $\mathsf{sk} \in \mathbb{Z}_q$ *and* $t \in [N]$*, each share for party* P_i *consists of a single partial secret key* $w_i \in \mathbb{Z}_q$*. We denote* $w_0 = \mathsf{sk}$*.*
2. *For every set* $S \subset [N]$ *with* $|S| = t$*, there exists an efficiently computable Lagrange coefficients* $\lambda_i^S \in \mathbb{Z}_q$ *such that*

$$w_0 = \sum_{i \in S} \lambda_i^S \cdot w_i.$$

4.2 TreeSSS from Shamir's Secret Sharing

In this section, we present the Tree Secret Sharing Scheme (TreeSSS), which is a novel linear secret sharing scheme designed to accommodate a large number of participants. TreeSSS is based on the well-known Shamir's secret sharing scheme, as well as classical results on threshold circuits from the literature [26, 28].

This new method will allow us to construct t-out-of-N threshold functions from Shamir's secret sharing for s-out-of-$(2s-1)$ threshold functions, where $s \ll N$.

We will describe an iterative algorithm, called TreeSS algorithm, for constructing TreeSSS and explain the key components that make it work. The TreeSS algorithm inputs Shamir's secret sharing scheme $\mathsf{SS}^{(1)}$ for s-out-of-$(2s-1)$ threshold functions, with $s \ll N$ and outputs $\ell = 2s-1$ secret shares. It also returns the iteration number L and the secret key sk and a set of secret shares.

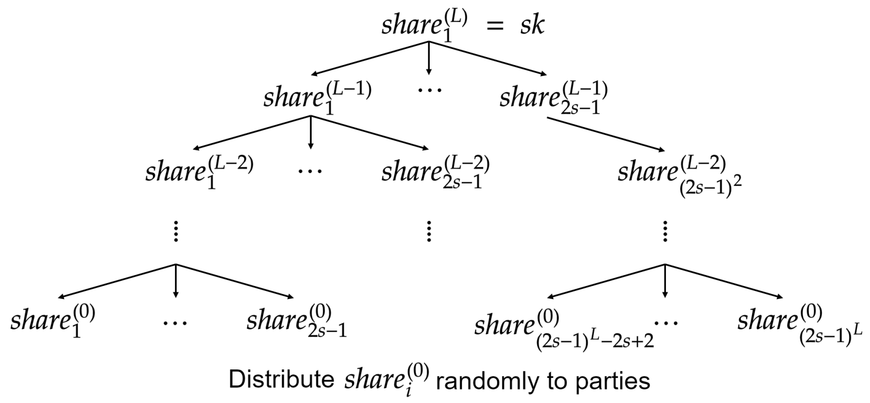

Fig. 4: High-level Overview of the TreeSS algorithm

The TreeSS algorithm works by applying the secret sharing scheme repeatedly in a tree-like manner. It starts by considering the secret key sk as a level-L secret share, $\mathsf{share}_1^{(L)}$. At each iteration, $L \geq i \geq 1$, the level-i secret shares, $\mathsf{share}_j^{(i)}$, are split into level-$(i-1)$ secret shares, $\{\mathsf{share}_k^{(i-1)}\}_{k\in\{\ell\cdot(j-1)+1,\cdots,\ell\cdot j\}}$. This process is repeated until level-0 secret shares, $\{\mathsf{share}_j^{(0)}\}_{j\in[\ell^L]}$, are obtained and distributed randomly to the parties. The detailed implementation of the TreeSS algorithm is provided in Algorithm 1.

Algorithm 1: TreeSS Algorithm

```
Input  : Shamir's secret sharing scheme for $s$-out-of-$(2s-1)$ threshold
         functions $\mathsf{SS}^{(1)}$, $\ell = (2s-1)$,
         The number of iterations $L$, secret key $\mathsf{sk}$
Output: Output the secret share for each $P_i$, $\{\mathsf{share}_j^{(0)}\}_{j\in T_i}$
1 $(\mathsf{share}_1^{(L-1)}, \cdots, \mathsf{share}_\ell^{(L-1)}) \leftarrow \mathsf{SS}^{(1)}(\mathsf{sk})$
2 for $i = L-1, \cdots, 1$ do
3     for $j = 1, \cdots, \ell^{L-i}$ do
4         $(\mathsf{share}_{\ell\cdot(j-1)+1}^{(i-1)}, \cdots, \mathsf{share}_{\ell\cdot(j-1)+\ell}^{(i-1)}) \leftarrow \mathsf{SS}^{(1)}(\mathsf{share}_j^{(i)})$
5     end for
6 end for
7 Distribute randomly $\{\mathsf{share}_j^{(0)}\}_{j\in[\ell^L]}$ to each party $P_i$
8 return The secret shares for each party $P_i$, $\{\mathsf{share}_j^{(0)}\}_{j\in T_i}$
```

Now, we introducing our new secret sharing scheme, the Tree Secret Sharing Scheme (TreeSSS), generated by the TreeSS algorithm. The basic unit secret sharing scheme used in the TreeSS algorithm is Shamir's secret sharing scheme $\mathsf{SS}^{(1)}$.

Definition 4.8 (L-TreeSSS) *Let $P = \{P_1, \cdots, P_N\}$ be a set of parties and $\mathsf{sk} \in \mathbb{Z}_q$ be a secret key. Given a Shamir's secret sharing scheme $\mathsf{SS}^{(1)}$ and iteration number L, a tree secret sharing scheme for iteration number L is a tuple of PPT algorithms L-$\mathsf{TreeSSS} = (\mathsf{TreeSSS.Share}, \mathsf{TreeSSS.Combine})$ that satisfies the following properties:*

- *The share algorithm $\mathsf{TreeSSS.Share}(\mathsf{sk}, \mathsf{SS}^{(1)}, L)$ takes as input a secret key sk, an Shamir's secret sharing $\mathsf{SS}^{(1)}$ and an iteration number L, and construct a new secret sharing scheme $\mathsf{SS}^{(L)}$ by TreeSS algorithm. Then, algorithm outputs $(\mathsf{sk}_1, \cdots, \mathsf{sk}_N)$, where $(\mathsf{sk}_1, \cdots, \mathsf{sk}_N)$ is a family of secret shares obtained by TreeSS algorithm.*
- *The combine algorithm $\mathsf{TreeSSS.Combine}(B)$ takes as input a set of shares $B = \{\mathsf{share}_j^{(0)}\}_{j\in\bigcup_{P_i\in S} T_i}$, where $T_i = \{j \mid P_i \text{ has a secret share } \mathsf{share}_j^{(0)}\}$. Then, combine algorithm computes $T^{(i)} = \{j \mid \mathsf{share}_j^{(i)} \text{ can be recovered by } B\}$ for index sets for $0 \leq i \leq L$, and $\hat{\mathsf{sk}}$ as follows:*

$$
\begin{aligned}
\hat{\mathsf{sk}} &= \sum_{j \in T^{(0)}} \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdots \lambda_{j_1}^{S_{j_1}^{(1)}} \cdot \lambda_{j}^{S_{j}^{(0)}} \cdot \mathsf{share}_j^{(0)} \\
&= \sum_{j_1 \in T^{(1)}} \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdots \lambda_{j_2}^{S_{j_2}^{(2)}} \cdot \lambda_{j_1}^{S_{j_1}^{(1)}} \cdot \mathsf{share}_{j_1}^{(1)} \\
&\vdots \\
&= \sum_{j_{L-2} \in T^{(L-2)}} \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \lambda_{j_{L-2}}^{S_{j_{L-2}}^{(L-2)}} \cdot \mathsf{share}_{j_{L-2}}^{(L-2)} \\
&= \sum_{j_{L-1} \in T^{(L-1)}} \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \mathsf{share}_{j_{L-1}}^{(L-1)} \\
&= \mathsf{share}^{(L)} = \mathsf{sk}
\end{aligned}
$$

where j_k is an index of level-k secret shares which uses $\mathsf{share}_j^{(0)}$ to recover itself , $S_{j_k}^{(k)}$ is a set of level-k secret shares including $\mathsf{share}_{j_k}^{(k)}$ which recover a level-$(k+1)$ secret share $\mathsf{share}_{j_{k+1}}^{(k+1)}$, and the Lagrange coefficient $\lambda_{j_k}^{S_{j_k}^{(k)}}$ are obtained from the Lagrange polynomial of Shamir's secret sharing $\mathsf{SS}^{(1)}$.
If a set of share B can not recover a level-L secret share, then the combine algorithm output $\perp$.

Our scheme can be used to construct an N-input majority function, just like the small-input majority gates studied in [28]. We make this connection by showing that Shamir's secret sharing can serve as a majority gate, and that iterative secret sharing can be seen as a form of secret sharing, as discussed in Section 2. Through the use of Lemma 3.7, we can prove that our scheme has a threshold structure of $\lceil \frac{N}{2} \rceil$-out-of-$N$ with appropriate iteration number L.

Theorem 4.9 *Let $P = \{P_1, \cdots, P_N\}$ be a set of parties and N be odd number. Given a Shamir's secret sharing $\mathsf{SS}^{(1)}$ with share matrix $\mathbf{V}_s$ and iteration number $L \geq \log_{c_s} N + \log_s N + O(1)$ where $c_s = \frac{2s-1}{2^{2s-2}} \cdot \binom{2s-2}{s-1}$, the L-TreeSSS satisfies the correctness and privacy with $\lceil \frac{N}{2} \rceil$-out-of-$N$ threshold structure and the number of secret shares is $(2s-1)^L = O(N^{\log_{c_s}(2s-1) + \log_s(2s-1)})$.*

Proof. First, we will prove L-TreeSSS satisfies the correctness with $\lceil \frac{N}{2} \rceil$-out-of-$N$ threshold structure and the number of secret shares is $O(N^{\log_{c_s}(2s-1) + \log_s(2s-1)})$.

Correctness. The correctness automatically holds underlying the following fact:

– As discussed in Section 2, the large input majority circuit $\mathsf{MAJ}_N : \{0,1\}^N \rightarrow \{0,1\}$ can be built from the small-input majority circuit MAJ_{2s-1} when the iteration number $L \geq ge \log_{c_s} N + \log_s N + O(1)$ where $c_s = \frac{2s-1}{2^{2s-2}} \cdot \binom{2s-2}{s-1}$ (Lemma 3.7).

– Shamir's secret sharing says that MAJ_{2s-1} corresponds to the share matrix MAJ_{2s-1}.

Therefore, the majority function constructed by using the majority gate MAJ_{2s-1} can be converted to a secret sharing scheme which satisfies the correctness of majority-threshold structure.

Furthermore, we conclude that Shamir's secret sharing with $\mathbf{V}_s$ distributes the secret value into $2s-1$ secret shares, and the total number of secret shares in the TreeSSS scheme would be $(2s-1)^L = O(N^{\log_{c_s}(2s-1)+\log_s(2s-1)}) \approx O(N^{3+2.3/\log s})$.

Privacy. Now, we demonstrate that the privacy of secret sharing (as defined in Definition 4.4) is held in our L-TreeSSS. Given a subset $S \subset P$ with $|S| < \lceil \frac{N}{2} \rceil$ and two secret keys $\mathsf{sk}_0, \mathsf{sk}_1$, we consider the following pairs of shares obtained by executing $\mathsf{TreeSSS.share}(\mathsf{sk}_b, \mathsf{SS}^{(1)}, L) \to (\mathsf{sk}_{b,1}, \cdots, \mathsf{sk}_{b,N})$ for $b \in 0, 1$:

$$\{\mathsf{sk}_{0,i}\}_{i \in S} \text{ and } \{\mathsf{sk}_{1,i}\}_{i \in S}$$

Our objective is to prove that these two pairs of shares are drawn from the same distribution.

In order to establish the privacy of TreeSSS, we will employ mathematical induction on the level L. For the base case of $L = 1$, we observe that TreeSSS is equivalent to Shamir's secret sharing scheme, which is known to satisfy the privacy of secret sharing. Hence, the two sets of secret shares $\{sk_{0,i}\}_{i \in S}, \{sk_{1,i}\}_{i \in S}$ follow the same distribution.

To continue the proof, we assume that a $k+1$-level TreeSSS, with each subtree corresponding to a k-level TreeSSS, satisfies the privacy of secret sharing. Our goal is to demonstrate that this property extends to the $(k+1)$-level TreeSSS.

For easy explanation, we define a family of level-i secret shares $S_b^{(i)}$ by

$$S_b^{(i)} = \{\mathsf{share}_{b,j}^{(i)} \mid \mathsf{share}_{b,j}^{(i)} \text{ can be recovered by } S\}$$

for every $i \in [L]$. By definition, $\{\mathsf{sk}_{b,i}\}_{i \in S} = S_b^{(0)}$, so we want to prove that $S_0^{(0)}$ and $S_1^{(0)}$ are indistinguishable. Since TreeSSS is iteratively constructed, we the level-1 secret shares $S_0^{(1)}, S_1^{(1)}$ can be viewed as the output of k-TreeSSS. Therefore, the assumption would be restated as $S_0^{(1)}, S_1^{(1)}$ having the same distribution.

To this end, $S_b^{(0)}$ can be divided into two subsets:

– $S_{b,P}^{(0)}$: a set of secret shares which *can* be used to recover $S_b^{(1)}$ secret shares.
– $S_{b,I}^{(0)}$: a set of secret shares which *cannot* be used to recover $S_b^{(1)}$ secret shares.

Note that by definition, level-0 secret shares $S_{b,P}^{(0)}$ come from level-1 secret shares in $S_b^{(1)}$ by using Shamir's secret sharing for every b. Since nobody can distinguish between $S_0^{(1)}, S_1^{(1)}$ by the assumption, $S_{0,P}^{(k)}, S_{1,P}^{(k)}$ also follow the identical distribution.

Moreover, $S_{0,I}^{(0)}, S_{1,I}^{(0)}$ comes from Shamir's secret sharing and cannot recover level-1 secret shares. Therefore they also follow the identical distribution because Shamir's secret sharing satisfies the privacy of secret sharing. Thus, $(k+1)$-TreeSSS satisfies the privacy because of $S_b^{(0)} = S_{b,P}^{(0)} \bigcup S_{b,I}^{(0)}$.

As a result, by the mathematical induction, we can conclude that L-TreeSSS satisfies the privacy for all positive integer L.

□

Remark 1. Our TreeSSS can use any other secret sharing scheme. Especially, if one use a basic unit secret sharing with the share matrix $\mathbf{D} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & -1 & -1 \end{bmatrix}^T$, one can construct the $\{0,1\}$-LSSS in [9]. We replace it with Shamir's secret sharing to make an efficient threshold FHE.

4.3 TreeSSS for t-out-of-N for arbitrary t

In this subsection, we explain the process of extending an $\lceil \frac{N}{2} \rceil$-out-of-$N$ TreeSSS to a t-out-of-N TreeSSS for any value of t. For odd values of N, $\lceil \frac{N}{2} \rceil$-out-of-$N$ TreeSSS can be generated using the TreeSS algorithm discussed in the previous section. To construct a TreeSSS for a t-out-of-N threshold function, we modify the number of parties N and the threshold t by following the iterative Shamir theorem outlined in Theorem 4.9.

Case 1) $t > \lceil \frac{N}{2} \rceil$. According to Theorem 4.9, a TreeSSS for a t-out-of-$(2t-1)$ threshold structure can be constructed for any value of t. However, since $2t-1 > N$, we disregard the secret shares beyond those distributed to the N parties.

Case 2) $t < \lceil \frac{N}{2} \rceil$. To achieve this, we construct a TreeSSS for $(t+r)$-out-of-$(N+r)$, where r satisfies $\frac{N+r+1}{2} = t+r$. In this case, $r = N - 2t + 1$ and $N + r$ is always odd. Using Theorem 4.9, we obtain a TreeSSS for $(t+r)$-out-of-$(N+r)$ similar to the previous example. We then distribute the secret shares to N parties and make the remaining r secret shares public. This can be treated as a TreeSSS for a t-out-of-N threshold structure.

Case 3) $t = \frac{N}{2}$ and N is even. It suffices to construct a TreeSSS for $(\frac{N}{2}+1)$-out-of-$N+1$ threshold structure. As we observed in the previous case, the existence of such a construction can be guaranteed by Theorem 4.9.

Remark that for each case, the total number of parties is less than $2N$. Therefore, the number of secret shares is still $O(N^{\log_{c_s}(2s-1)+\log_s(2s-1)})$ with constant integer s.

5 Theshold Fully Homomorphic Encryption

5.1 Definitions

This section presents the definitions and properties of the threshold fully homomorphic encryption. We follow presentations of the original paper [9].

Definition 5.1 (Threshold Fully Homomorphic Encryption (TFHE)) *Let λ be the security parameter and d be a depth bound. Let $P = \{P_1, \cdots, P_N\}$ be a set of parties, and let $\mathbb{A}_t$ be a threshold structures on P. A threshold fully homomorphic encryption scheme for $\mathbb{A}_t$ is a tuple of PPT algorithms $\mathsf{TFHE} = (\mathsf{TFHE.Setup}, \mathsf{TFHE.Enc}, \mathsf{TFHE.Eval}, \mathsf{TFHE.PartDec}, \mathsf{TFHE.FinDec})$ that satisfies the following properties:*

- *The setup algorithm $\mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_t)$ takes as input the security parameter λ, a depth bound d, and a threshold structure $\mathbb{A}$, and outputs $(\mathsf{pk}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N)$, where pk is a public key and $\{\mathsf{sk}_i\}$ is a set of secret shares.*
- *The encryption algorithm $\mathsf{TFHE.Enc}(\mathsf{pk}, \mu)$ takes as input a public key pk and a message $\mu \in \{0, 1\}$, and outputs ciphertext ct.*
- *The evaluation algorithm $\mathsf{TFHE.Eval}(C, \mathsf{ct}_1, \cdots, \mathsf{ct}_l, \mathsf{pk})$ takes as input a circuit of which depth is less than or equal d, a tuple of ciphertexts $\mathsf{ct}_1, \cdots, \mathsf{ct}_l$ and a public key pk, and outputs an evaluated ciphertext $\hat{\mathsf{ct}}$.*
- *The partial decryption algorithm $\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \hat{\mathsf{ct}})$ takes as input a public key pk, a secret key share sk_i and the ciphertext $\hat{\mathsf{ct}}$ and outputs a partial decryption p_i related to the party P_i.*
- *The final decryption algorithm $\mathsf{TFHE.FinDec}(\mathsf{pk}, B)$ take as input a public key pk, and a set $B = \{p_i\}_{i \in S}$ for some $S \subset P$, and outputs a message $\hat{\mu} \in \{0, 1, \bot\}$.*

Hereafter, we use notations from Definition 5.1.

Definition 5.2 (Correctness of Evaluation) *We say TFHE scheme is correct if for any evaluated ciphertext $\hat{\mathsf{ct}}$ generated by $\mathsf{TFHE.Eval}(C, \mathsf{ct}_1, \cdots, \mathsf{ct}_l, \mathsf{pk})$ satisfies*

$$\Pr[\mathsf{FinDec}(\mathsf{pk}, \{\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \hat{\mathsf{ct}})\}_{i \in S}) = C(\mu_1, \cdots, \mu_l)] = 1 - \mathsf{negl}(\lambda).$$

Definition 5.3 (Compactness) *We say TFHE scheme is compact if for any ciphertext ct generated from the algorithm of $\mathsf{TFHE.Enc}$ and the partial decryption p_i obtained by $\mathsf{TFHE.PartDec}$, there are polynomials $\mathsf{poly}_1, \mathsf{poly}_2$ such that for any $j \in [N]$, it holds that*

$$|\mathsf{ct}| \leq \mathsf{poly}_1(\lambda, d) \text{ and } |p_i| \leq \mathsf{poly}_2(\lambda, d, N).$$

TFHE requires two types of security notions. One is the semantic security for encryption algorithm, and the simulation security is needed for partial decryption.

Definition 5.4 (Semantic security) *Given the security parameter λ and a depth bound d, for any PPT adversary $\mathcal{A}$, the following experiment $Expt_{\mathcal{A}, \mathsf{TFHE}}(1^\lambda, 1^d)$ outputs 1 with $\frac{1}{2}$ probability except for negligible probability:*

$Expt_{\mathcal{A}, \mathsf{TFHE}}(1^\lambda, 1^d)$:

1. *For every security parameter λ and a depth bound d, the adversary $\mathcal{A}$ outputs a threshold structure $\mathbb{A}_t$ where $1 \le t \le N$.*
2. *The challenger $\mathcal{C}$ runs $\mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_t) \to (\mathsf{pk}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N)$, and gives pk to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs a set $S \subset \{P_1, \cdots, P_N\}$ such that $S \notin \mathbb{A}_t$.*
4. *The challenger runs $\mathsf{TFHE.Enc}(\mathsf{pk}, b) \to \mathsf{ct}$ and provides $\{\mathsf{ct}, \{\mathsf{sk}_i\}_{i\in S}\}$ to $\mathcal{A}$.*
5. *$\mathcal{A}$ outputs a guess b'.*
6. *The experiment outputs 1 if $b = b'$.*

Definition 5.5 (Simulation Security) *For any security parameter λ, a depth bound d, and a threshold structure $\mathbb{A}_t$, the following holds. There exists a stateful PPT algorithm $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for any PPT adversary $\mathcal{A}$, the following experiments $Expt_{\mathcal{A},Real}(1^\lambda, 1^d)$ and $Expt_{\mathcal{A},Ideal}(1^\lambda, 1^d)$ are indistinguishable:*

$Expt_{\mathcal{A},Real}(1^\lambda, 1^d)$:

1. *For every security parameter λ and a depth bound d, the adversary $\mathcal{A}$ outputs a threshold structure $\mathbb{A}_t$ where $1 \le t \le N$.*
2. *The challenger $\mathcal{C}$ runs $\mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_t) \to (\mathsf{pk}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N)$, and gives pk to $\mathcal{A}$*
3. *$\mathcal{A}$ outputs a maximal invalid set $S^* \subset \{P_1, \cdots, P_N\}$ and messages $\mu_1, \cdots, \mu_k \in \{0, 1\}$.*
4. *$\mathcal{C}$ provides a family of key shares and ciphertexts $\{\{\mathsf{sk}_i\}_{i\in S^*}, \{\mathsf{TFHE.Enc}(\mathsf{pk}, \mu_i)\}_{i\in[k]}\}$ to $\mathcal{A}$.*
5. *$\mathcal{A}$ issues a polynomial number of adaptive queries of the form $(S \subset \{P_1, \cdots, P_N\}, C)$ for circuits $C : \{0, 1\}^k \to \{0, 1\}$ of depth at most d. For each query, $\mathcal{C}$ computes $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \cdots, \mathsf{ct}_k)$ and provides $\{\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \hat{\mathsf{ct}})\}_{i\in S}$ to $\mathcal{A}$.*
6. *At the end of the experiment, $\mathcal{A}$ outputs a distinguishing bit b.*

$Expt_{\mathcal{A},Ideal}(1^\lambda, 1^d)$:

1. *Same as the first step of $Expt_{\mathcal{A},Real}(1^\lambda, 1^d)$*
2. *The challenger $\mathcal{C}$ runs $\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A}_t) \to (\mathsf{pk}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N, \mathsf{st})$, and gives pk to $\mathcal{A}$.*
3. *Same as the 3rd step of $Expt_{\mathcal{A},Real}(1^\lambda, 1^d)$*
4. *Same as the 4th step of $Expt_{\mathcal{A},Real}(1^\lambda, 1^d)$*
5. *$\mathcal{A}$ issues a polynomial number of adaptive queries of the form $(S \subset \{P_1, \cdots, P_N\}, C)$, where $C : \{0, 1\}^k \to \{0, 1\}$ is a circuit of depth at most d. For each query, $\mathcal{C}$ runs the simulator*

$$\{\mathcal{S}_2(C, \{\mathsf{ct}_1, \cdots, \mathsf{ct}_k\}, C(\mu_1, \cdots, \mu_k), S, \mathsf{st}) \to \{p_i\}_{i\in S}$$

   and sends $\{p_i\}_{i\in S}$ to $\mathcal{A}$.
6. *At the end of the experiment, $\mathcal{A}$ outputs a distinguishing bit b.*

5.2 TFHE using TreeSSS

Let $P = \{P_1, \cdots, P_N\}$ be a set of parties. Then, the communication efficient TFHE can be built from the following primitives:

- Let FHE be a special fully homomorphic encryption scheme (Definition 3.12) with noise bound B and multiplicative constant $((2s-1)!)^L$ where $L \geq \log_{c_s} N + \log_s N + O(1)$ and c_s is $\frac{2s-1}{4^{s-1}} \cdot \binom{2s-2}{s-1}$ for a positive integer $s \geq 2$.
- Let SS be a L-TreeSSS with Vandermonde matrix $\mathbf{V}_s \in \mathbb{Z}_q^{(2s-1)\times s}$ (Section 4.2).

The construction presented in this paper is similar to the one in [9], except that we utilize a TreeSSS with a Vandermonde matrix as opposed to a $\{0,1\}$-LSSS instantiated by [29]. As a result, most of the security proofs are similar in both cases, with the exception of Theorem 5.10, which forms the core of this paper. Consequently, we only include the proof for this theorem in the main text, while the remaining proofs can be found in the Supplementary material.

Construction 5.6 We can construct a tuple of PPT algorithms as follows:

- $(\mathsf{pk}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_t)$:
  1. Sample $(\mathsf{fhepk}, \mathsf{fhesk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$.
  2. Compute $(\mathsf{share}_1^{(0)}, \cdots, \mathsf{share}_{(2s-1)^L}^{(0)}) \leftarrow \mathsf{SS.Share}(\mathsf{fhesk}, \mathbb{A}_t)$ where SS a L-TreeSSS with $\mathbf{V}_s$ in Section 4.2.
  3. Distribute secret shares uniform randomly to each party and define a index set of each party $T_i := \{j \mid P_i \textit{ has } \mathsf{share}_j^{(0)}\}$.
  4. Return $\mathsf{pk} = \mathsf{fhepk}$ and $\mathsf{sk}_i = \{\mathsf{share}_j^{(0)}\}_{j\in T_i}$ for $i \in [N]$.
- $\mathsf{ct} \leftarrow \mathsf{TFHE.Enc}(\mathsf{pk}, \mu)$: Sample $\mathsf{ct} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, \mu)$ and return ct.
- $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(C, \mathsf{ct}_1, \cdots, \mathsf{ct}_k, \mathsf{pk})$: Compute $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(C, \mathsf{ct}_1, \cdots, \mathsf{ct}_k, \mathsf{pk})$ and return $\hat{\mathsf{ct}}$.
- $p_i \leftarrow \mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \hat{\mathsf{ct}})$:
  1. Sample a noise flooding error error $e_j \leftarrow [-B_{sm}, B_{sm}]$ and compute
  $$\hat{\mathbf{p}}_j^{(0)} = \mathsf{FHE.Dec}_0(\mathsf{share}_j^{(0)}, \mathsf{ct}) + ((2s-1)!)^L e_j \in \mathbb{Z}_q$$
  for every $j \in T_i$.
  2. Return $p_i = \{\hat{\mathbf{p}}_j^{(0)}\}_{j\in T_i}$ as its partial decryption.
- $\hat{\mu} \leftarrow \mathsf{TFHE.FinDec}(\mathsf{pk}, B)$:
  1. Check if $S \in \mathbb{A}_t$ or not: If $S \notin \mathbb{A}_t$, return $\perp$.
  2. If $S \in \mathbb{A}_t$, compute a minimal valid share set $T \subset \cup_{i\in S} T_i$ and $\mu \leftarrow \mathsf{FHE.Dec}_1(\sum_{j\in T} c_j^S \hat{\mathbf{p}}_j^{(0)})$.
  3. Return $\hat{\mu}$.

Theorem 5.7 *([9]) Suppose FHE is a compact fully homomorphic encryption scheme. Then, the TFHE scheme from Construction 5.6 satisfies compactness.*

Theorem 5.8 *Suppose* FHE *is a special fully homomorphic encryption scheme that satisfies correctness with noise bound B and* SS *is a level-i_s Shamir's secret sharing scheme that satisfies correctness. Then, the* TFHE *scheme from Construction 5.6 with parameter B_{sm} such that $B+((2s-1)!)^{2L}\cdot(2s-1)^L\cdot B_{sm} \leq \lfloor \frac{q}{4} \rceil$ satisfies correctness where $L \geq \log_{c_s} N + \log_s N + O(1)$.*

Theorem 5.9 *([9]) Suppose* FHE *is a fully homomorphic encryption scheme that satisfies security and* SS *is a secret sharing scheme that satisfies privacy. Then, the* TFHE *scheme from Construction 5.6 satisfies semantic security.*

Theorem 5.10 *Suppose* FHE *is a fully homomorphic encryption scheme that satisfies security and* SS *is a secret sharing scheme that satisfies correctness and privacy. Then, the* TFHE *scheme from Construction 5.6 with parameter B_{sm} such that $B \cdot ((2s-1)!)^L / B_{sm} = \mathsf{negl}(\lambda)$ satisfies simulation security where $L \geq \log_{c_s} N + \log_s N + O(1)$.*

Proof (of Theorem 5.10). We adapt the security proof in [9] according to the our construction. We define a series of the hybrid experiments between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- $\mathbf{H}_0$: This is a real experiment $Expt_{\mathcal{A},Real}(1^\lambda, 1^d)$ of TFHE in Definition 5.5.
- $\mathbf{H}_1$: Same as $\mathbf{H}_0$ except for $\mathcal{C}$ simulates the partial decryption for $\mathcal{A}$'ss queries. More precisely, $\mathcal{C}$ first computes the maximal invalid secret shares $\{\mathsf{share}_j\}_{j\in T^*}$ where T^* is the union of all T_i for $i \in S^*$. Then, $\mathcal{C}$ can obtain the partial decryption $\mathsf{TFHE.PartDec}(\mathsf{pk}, \hat{\mathsf{ct}}, \mathsf{sk}_i)$ for $i \in S$ by using $\{\mathsf{share}_j\}_{j\in T^*}$ and $C(\mu_1, \cdots, \mu_k)$ for each query (S, C). The partial decryption algorithm takes in $(\mathsf{pk}, \hat{\mathsf{ct}}, \mathsf{sk}_i)$ and outputs $p_i = \{\hat{\mathbf{p}}_j^{(0)}\}_{j\in T_i}$, based on the following conditions:

  (Case 1) $j \in T^*$: In this case, $\mathcal{C}$ already has share_j, so $\mathcal{C}$ can compute $\hat{\mathbf{p}}_j^{(0)}$ as follows:

  $$\hat{\mathbf{p}}_j^{(0)} = \mathsf{FHE.Dec}_0(\mathsf{share}_j, \hat{\mathsf{ct}}) + ((2s-1)!)^L \cdot e_j,$$

  where e_j is uniformly sampled from $[-B_{sm}, B_{sm}]$.

  (Case 2) $j \notin T^*$: By definition of T^* that is a maximal invalid secret shares, $\bar{T} = T^* \cup \{j\}$ should be a set of valid shares. Hence, there are multiples of Lagrange coefficients for each $k \in \bar{T}$ such that $\sum_{k\in\bar{T}} c_k \cdot \mathsf{share}_k = \mathsf{fhesk}$. Then, $\mathcal{C}$ returns

  $$\hat{\mathbf{p}}_j^{(0)} = (c_j)^{-1}\cdot C(\mu_1, \cdots, \mu_k)\cdot \frac{q}{2} - \sum_{j'\in T^*} (c_j)^{-1}\cdot \mathsf{FHE.Dec}_0(\mathsf{share}_{j'}, \hat{\mathsf{ct}}) + ((2s-1)!)^L \cdot e_j$$

  where e_j is uniformly sampled from $[-B_{sm}, B_{sm}]$.
- $\mathbf{H}_2$: Same as $\mathbf{H}_1$ except that $\mathcal{C}$ randomly samples sk_i. This is an ideal experiment $Expt_{\mathcal{A},ideal}(1^\lambda, 1^d)$ of TFHE in Definition 5.5.

Now, we will prove that hybrid experiments, $\mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2$, are statistical indistinguishable.

Lemma 5.11 $\mathbf{H}_0 \approx_s \mathbf{H}_1$

Proof (of Lemma 5.11). The only difference between $\mathbf{H}_0$ and $\mathbf{H}_1$ is an algorithm of partial decryption $\hat{\mathbf{p}}_j^{(0)}$ for $j \notin T^*$. Due to the correctness of FHE and definition of special FHE, it holds that $\frac{q}{2} \cdot C(\mu_1, \cdots, \mu_k) = \mathsf{FHE.Dec}_0(\mathsf{sk}, \hat{\mathsf{ct}}) + \tilde{e}$ where an error $\tilde{e}$ is sampled uniformly at random in $[-\mathbf{c}B, \mathbf{c}B]$ and $\mathbf{c} = ((2s-1)!)^L$.

As a result, we can reinterpret $\hat{\mathbf{p}}_j^{(0)}$

$$
\begin{aligned}
\hat{\mathbf{p}}_j^{(0)} &= (c_j)^{-1} C(\mu_1, \cdots, \mu_k) \cdot \frac{q}{2} - \sum_{j' \in T^*} (c_j)^{-1} \mathsf{FHE.Dec}_0(\mathsf{share}_{j'}, \hat{\mathsf{ct}}) + \mathbf{c} \cdot e_j \\
&= (c_j)^{-1} (\mathsf{FHE.Dec}_0(\mathsf{sk}, \hat{\mathsf{ct}}) + \tilde{e}) - \sum_{j' \in T^*} (c_j)^{-1} \mathsf{FHE.Dec}_0(\mathsf{share}_{j'}, \hat{\mathsf{ct}}) + \mathbf{c} \cdot e_j \\
&= \mathsf{FHE.Dec}_0 \left((c_j)^{-1} \cdot \left(\mathsf{sk} - \sum_{j' \in T^*} \mathsf{share}_{j'} \right), \hat{\mathsf{ct}} \right) + (c_j)^{-1} \cdot \tilde{e} + \mathbf{c} \cdot e_j \\
&= \mathsf{FHE.Dec}_0(\mathsf{share}_j, \hat{\mathsf{ct}}) + (c_j)^{-1} \cdot \tilde{e} + \mathbf{c} \cdot e_j.
\end{aligned}
$$

In partial decryption in $\mathbf{H}_1$, there is an extra error term $(c_j)^{-1} \cdot \tilde{e}$. Since $\tilde{e}$ is a multiple of $((2s-1)!)^L$ and c_j is a multiple of Lagrange coefficient, it follows from Lemma 2.3 that $|(c_j)^{-1} \cdot \tilde{e}| \le ((2s-1)!)^{2L} \cdot B$. The bound B_{sm} satisfies $(B \cdot ((2s-1)!)^L)/B_{sm} = \mathsf{negl}(\lambda)$, making the experiments $\mathbf{H}_0$ and $\mathbf{H}_1$ indistinguishable due to the noise flooding technique (Lemma 3.1).

□

Lemma 5.12 $\mathbf{H}_1 \approx_s \mathbf{H}_2$

Proof (of Lemma 5.12). The difference between $\mathbf{H}_1$ and $\mathbf{H}_2$ lies in the method of sampling the secret keys $\{\mathsf{sk}_i\}_{i \in S^*}$, where S^* is an invalid set. The privacy of the secret sharing scheme ensures that no party can distinguish between the two distributions of secret keys for any invalid set. Therefore, an adversary cannot distinguish between $\mathbf{H}_1$ and $\mathbf{H}_2$ if the secret sharing scheme provides the desired privacy. □

Lemma 5.11 and Lemma 5.12 says that $\mathbf{H}_0$ is also statistically indistinguishable to $\mathbf{H}_2$. As a result, Construction 5.6 achieves the simulation security. □

6 Communication Efficient Universal Thresholdizer

As shown in Table 2, our TFHE shows superiority over previous compact TFHE in terms of small share key sizes. This translates to lower communication costs during partial decryption.

Building a communication-efficient universal thresholdizer can then be achieved by combining our TFHE with other primitives, as proven by [9] through the following theorems. Throughout this section, we adopt the definitions and theorems from [9] that provides the concept of UT and the first construction.

6.1 Definition

This section presents definitions and properties of the universal thresholdizer. We basically follow presentations of the original paper [9].

Definition 6.1 (Universal Thresholdizer [9]) *Let $P = \{P_1, \cdots, P_N\}$ be a set of parties and $\mathbb{A}_t$ be a threshold structures on P. A universal thresholdizer scheme for $\mathbb{A}_t$ is a tuple of PPT algorithms $\mathsf{UT} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ such that*

- *The setup algorithm of UT, $\mathsf{UT.Setup}(1^\lambda, d, \mathbb{A}_t, \mathbf{x})$ takes as input the security parameter λ, a depth bound d, a threshold structure $\mathbb{A}_t$ and a message $\mathbf{x} \in \{0,1\}^k$ and returns a public parameter pp and a family of shares $\{\mathsf{sk}_i\}_{i\in[N]}$.*
- *The evaluation algorithm of UT, $\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, C)$ takes as input the public parameter pp, a share sk_i, and a depth d circuit $C : \{0,1\}^k \to \{0,1\}$, and outputs a partial evaluation, say y_i.*
- *The verification algorithm of UT, $\mathsf{UT.Verify}(\mathsf{pp}, y_i, C)$ takes as input the public parameter pp, a partial evaluation y_i, and a circuit C and returns 0 (accept) or 1 (reject).*
- *The combining algorithm of UT, $\mathsf{UT.Combine}(\mathsf{pp}, \{y_i\}_{i\in S})$ takes as input the public parameter, and a set of partial evaluations y_i, and returns the final evaluation y.*

Hereafter, we use notations from Definition 6.1.

Definition 6.2 (Compactness) *We say that UT scheme is compact if there is a polynomial poly such that for any $i \in [N]$, $|y_i| \le \mathsf{poly}(\lambda, d, N)$, where $y_i \leftarrow \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, C)$.*

Definition 6.3 (Evaluation Correctness) *We say UT scheme satisfies the correct of evaluation if the following holds:*

$$\Pr[\mathsf{UT.Combine}(\mathsf{pp}, \{\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, C)\}_{i\in S}) = C(x)] = 1 - \mathsf{negl}(\lambda)$$

for any $(\mathsf{pp}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}_t, \mathbf{x})$.

Definition 6.4 (Verification Correctness) *We say UT scheme satisfies the correct of verification if the following holds*

$$\Pr[\mathsf{UT.Verify}(\mathsf{pp}, y_i, C) = 1] = 1$$

for any $(\mathsf{pp}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}_t, \mathbf{x})$ and $y_i \leftarrow \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, C)$.

Definition 6.5 (Robustness) *A UT holds robustness if for any λ and d, the following satisfies: for any PPT adversary $\mathcal{A}$, the experiment*

$Expt_{\mathcal{A},\mathsf{UT},\mathsf{robustness}}(1^\lambda, 1^d)$:

1. *Given λ and d, the adversary $\mathcal{A}$ returns a message $\mathbf{x} \in \{0,1\}^k$ and a threshold structure $\mathbb{A}_t$*

2. *The challenger $\mathcal{C}$ runs $(\mathsf{pp}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}_t, \mathbf{x})$ and provides $(\mathsf{pp}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ a fake partial evaluation y_i^*.*
4. *$\mathcal{C}$ returns 1 if $y_i^* \neq \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, C)$ and $\mathsf{UT.Verify}(\mathsf{pp}, y_i^*, C) = 1$.*

Definition 6.6 (Security) *Given the security parameter λ, a depth bound d, and threshold structure $\mathbb{A}_t$, the following holds. There exists a stateful PPT algorithm $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for any PPT adversary $\mathcal{A}$, the following experiments $Expt_{\mathcal{A},Real}(1^\lambda, 1^d)$ and $Expt_{\mathcal{A},Ideal}(1^\lambda, 1^d)$ are distinguishable:*

$Expt_{\mathcal{A},Real}(1^\lambda, 1^d)$:
1. *On input the security parameter λ and a depth bound d, the adversary $\mathcal{A}$ outputs a threshold structure $\mathbb{A}_t$ and a message $\mathbf{x} \in \{0,1\}^k$.*
2. *The challenger runs $\mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}_t, \mathbf{x}) \rightarrow (\mathsf{pp}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N)$, and provides pp to $\mathcal{A}$*
3. *$\mathcal{A}$ outputs a maximal invalid set $S^* \subset \{P_1, \cdots, P_N\}$.*
4. *The challenger provides the key shares $\{\mathsf{sk}_i\}_{i \in S^*}$ to $\mathcal{A}$.*
5. *The adversary $\mathcal{A}$ issues a polynomial number of adaptive queries of the form $(S \subset \{P_1, \cdots, P_N\}, C)$ for circuits $C : \{0,1\}^k \rightarrow \{0,1\}$ of depth at most d. For each query, the challenger provides $\{\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, C) \rightarrow y_i\}_{i \in S}$ to $\mathcal{A}$.*
6. *At the end of the experiment, $\mathcal{A}$ outputs a distinguishing bit b.*

$Expt_{\mathcal{A},Ideal}(1^\lambda, 1^d)$:
1. *On input the security parameter λ and a depth bound d, the adversary $\mathcal{A}$ outputs a threshold structure $\mathbb{A}_t$ and a message $\mathbf{x} \in \{0,1\}^k$.*
2. *The challenger runs $\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A}_t) \rightarrow (\mathsf{pp}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N, \mathsf{st})$, and provides pp to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs a maximal invalid set $S^* \subset \{P_1, \cdots, P_N\}$.*
4. *The challenger provides the key shares $\{\mathsf{sk}_i\}_{i \in S^*}$ to $\mathcal{A}$.*
5. *The adversary $\mathcal{A}$ issues a polynomial number of adaptive queries of the form $(S \subset \{P_1, \cdots, P_N\}, C)$ for circuits $C : \{0,1\}^k \rightarrow \{0,1\}$ of depth at most d. For each query, the challenger runs the simulator $\{\mathcal{S}_2(\mathsf{pp}, C, C(\mathbf{x}), S, \mathsf{st}) \rightarrow \{y_i\}_{i \in S}$ and sends $\{y_i\}_{i \in S}$ to $\mathcal{A}$.*
6. *At the end of the experiment, $\mathcal{A}$ outputs a distinguishing bit b.*

6.2 Construction

We recall the construction of universal thresholdizer from TFHE, and non-interactive zero knowledge proof system with pre-processing (Definition 3.13) [30, 38] and non-interactive commitment (Definition 3.15) scheme [6]. We again note that the description is based on the original paper [9].

Construction 6.7 *We can construct a tuple of PPT algorithms as follows:*

- $\mathsf{UT.Setup}(1^\lambda, d, \mathbb{A}_t, \mathbf{x})$:
  1. *Sample TFHE keys $(\mathsf{tfhepk}, \{\mathsf{tfhesk}\}_{i \in [N]}) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_t)$.*

2. *Sample* $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Enc}(\mathsf{tfhepk}, x_i)$ *for* $i = 1, \dots, k$.
3. *Generate reference strings* $(\sigma_{V,i}, \sigma_{P,i}) \leftarrow \mathsf{PZK.Pre}(1^\lambda)$*, commitment randomness* $\mathbf{r}_i \leftarrow \{0,1\}^\lambda$ *and commitments* $\mathsf{com}_i \leftarrow \mathsf{C.Com}(\mathsf{tfhesk}_i, \mathbf{r}_i)$ *for* $i = 1, \dots, N$.
4. *Return* pp *and* sk *as follows:*

$$\mathsf{pp} = \left(\mathsf{tfhepk}, \{\mathsf{ct}_i\}_{i\in[k]}, \{\sigma_{V,i}\}_{i\in[N]}, \{\mathsf{com}_i\}_{i\in[N]}\right), \mathsf{sk}_i = \{\mathsf{tfhesk}_i, \sigma_{P,i}, \mathbf{r}_i\}_{i\in[N]}$$

- $\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, C)$:
  1. *Compute* $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{tfhepk}, C, \mathsf{ct}_1, \mathsf{ct}_2, \dots, \mathsf{ct}_k)$
  2. *Compute* $\mathbf{p}_i \leftarrow \mathsf{TFHE.PartDec}(\mathsf{tfhepk}, \hat{\mathsf{ct}}, \mathsf{tfhesk}_i)$
  3. *Construct a statement* $\Psi_i = \Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathbf{p}_i)$ *that has the following relation*

     $$\exists\,(\mathsf{tfhesk}_i, \mathbf{r}_i) : \mathsf{com}_i = \mathsf{C.Com}(\mathsf{tfhesk}_i; \mathbf{r}_i) \wedge \mathbf{p}_i = \mathsf{TFHE.PartDec}(\mathsf{pp}, \hat{\mathsf{ct}}, \mathsf{tfhesk}_i)$$

  4. *Generate a NIZK proof* $\pi_i \leftarrow \mathsf{PZK.Prove}(\sigma_{P,i}, \Psi_i, (\mathsf{tfhesk}_i, \mathbf{r}_i))$
  5. *Return* $y_i = (\mathbf{p}_i, \pi_i)$
- $\mathsf{UT.Verify}(\mathsf{pp}, y_i, C)$: *Parse* $y_i = (\mathbf{p}_i, \pi_i)$ *and construct a statement* $\Psi_i = \Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathbf{p}_i)$ *and return the result of*

  $$\mathsf{PZK.Verify}(\sigma_{V,i}, \Psi_i, \pi_i).$$

- $\mathsf{UT.Combine}(\mathsf{pp}, \{y_i\}_{i\in S})$: *Parse* $y_i = (\mathbf{p}_i, \pi_i)$ *and return the result of*

  $$\mathsf{TFHE.FinDec}(\mathsf{tfhepk}, \{\mathbf{p}_i\}_{i\in S}).$$

Consequently, we directly apply the following theorem. For more details, we refer [9].

Theorem 6.8 ([9]) *Suppose that there are cryptographic schemes that satisfies the following:*

- *threshold fully homomorhpic encryption that satisfies compactness (Definition 5.3), correctness of evaluation (Definition 5.2), semantic security (Definition 5.4) and simulation security (Definition 5.5).*
- *zero knowledge proof system with pre-processing which satisfies zero-knowledge and soundness.*
- *non-interactive commitment scheme that holds perfect binding and computational hiding.*

Then, Construction 6.7 is an universal thresholdizer scheme such that compactness (Definition 6.2), evaluation correctness (Definition 6.3), verification correctness (Definition 6.4), robustness (Definition 6.5) and security (Definition 6.6).

References

1. Shweta Agrawal, Xavier Boyen, Vinod Vaikuntanathan, Panagiotis Voulgaris, and Hoeteck Wee. Functional encryption for threshold functions (or fuzzy ibe) from lattices. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography – PKC 2012*, pages 280–297, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
2. Shweta Agrawal, Damien Stehle, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. *Cryptology ePrint Archive*, 2022.
3. Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 483–501. Springer, 2012.
4. Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. Secure mpc: Laziness leads to god. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020*, pages 120–150, Cham, 2020. Springer International Publishing.
5. Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. *Journal of Cryptology*, 33(2):459–493, 2020.
6. Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 15(1):23–27, 1983.
7. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *International Workshop on Public Key Cryptography*, pages 31–46. Springer, 2003.
8. Dan Boneh, Rosario Gennaro, and Steven Goldfeder. Using level-1 homomorphic encryption to improve threshold dsa signatures for bitcoin wallet security. In *International Conference on Cryptology and Information Security in Latin America*, pages 352–377. Springer, 2017.
9. Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter MR Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *Annual International Cryptology Conference*, pages 565–596. Springer, 2018.
10. Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from lwe with polynomial modulus. *Cryptology ePrint Archive*, 2023.
11. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.
12. Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key fhe with short ciphertexts. In *Annual Cryptology Conference*, pages 190–213. Springer, 2016.
13. Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. Uc non-interactive, proactive, threshold ecdsa with identifiable aborts. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1769–1787, 2020.
14. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security*, pages 409–437. Springer, 2017.

15. Jung Hee Cheon, Dongwoo Kim, and Duhyeong Kim. Efficient homomorphic comparison methods with optimal complexity. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020*, pages 221–256, Cham, 2020. Springer International Publishing.
16. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *international conference on the theory and application of cryptology and information security*, pages 3–33. Springer, 2016.
17. Siddhartha Chowdhury, Sayani Sinha, Animesh Singh, Shubham Mishra, Chandan Chaudhary, Sikhar Patranabis, Pratyay Mukherjee, Ayantika Chatterjee, and Debdeep Mukhopadhyay. Efficient fhe with threshold decryption and application to real-time systems. *Cryptology ePrint Archive*, 2022.
18. Ivan Damgård and Maciej Koprowski. Practical threshold rsa signatures without a trusted dealer. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 152–165. Springer, 2001.
19. Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 522–533. ACM, 1994.
20. Yvo Desmedt and Yair Frankel. Threshold cryptosystesns. *Advance in Cryptology*, pages 305–315, 1989.
21. Yair Frankel. A practical protocol for large group oriented networks. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 56–61. Springer, 1989.
22. Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ecdsa with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1179–1194, 2018.
23. Rosario Gennaro and Steven Goldfeder. One round threshold ecdsa with identifiable abort. *Cryptology ePrint Archive*, 2020.
24. Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security. In *International Conference on Applied Cryptography and Network Security*, pages 156–174. Springer, 2016.
25. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold dss signatures. *Information and Computation*, 164(1):54–84, 2001.
26. Oded Goldreich. *On (Valiant's) Polynomial-Size Monotone Formula for Majority*, pages 17–23. Springer International Publishing, Cham, 2020.
27. S Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round mpc with fairness and guarantee of output delivery. In *Annual Cryptology Conference*, pages 63–82. Springer, 2015.
28. Arvind Gupta and Sanjeev Mahajan. Using amplification to compute majority with small majority gates. *Computational Complexity*, 6(1):46–63, 1996.
29. Aayush Jain, Peter MR Rasmussen, and Amit Sahai. Threshold fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2017:257, 2017.
30. Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *Conference on the Theory and Application of Cryptography*, pages 353–365. Springer, 1990.
31. Yongwoo Lee, Daniele Micciancio, Andrey Kim, Rakyong Choi, Maxim Deryabin, Jieun Eom, and Donghoon Yoo. Efficient fhew bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. *Cryptology ePrint Archive*, 2022.

32. Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 568–588. Springer, 2011.
33. Yehuda Lindell. Fast secure two-party ecdsa signing. In *Annual International Cryptology Conference*, pages 613–644. Springer, 2017.
34. Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234, 2012.
35. Philip MacKenzie and Michael K Reiter. Two-party generation of dsa signatures. *International Journal of Information Security*, 2(3):218–239, 2004.
36. Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key fhe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 735–763. Springer, 2016.
37. Chris Peikert and Sina Shiehian. Multi-key fhe from lwe, revisited. In *Theory of Cryptography Conference*, pages 217–238. Springer, 2016.
38. Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge with preprocessing. In *Conference on the Theory and Application of Cryptography*, pages 269–282. Springer, 1988.
39. Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
40. Victor Shoup. Practical threshold signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 207–220. Springer, 2000.
41. Douglas R Stinson and Reto Strobl. Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates. In *Australasian Conference on Information Security and Privacy*, pages 417–434. Springer, 2001.
42. Leslie G. Valiant. Short monotone formulae for the majority function. *Journal of Algorithms*, 5(3):363–366, 1984.
43. Zhedong Wang, Xiong Fan, and Feng-Hao Liu. Fe for inner products and its application to decentralized abe. In Dongdai Lin and Kazue Sako, editors, *Public-Key Cryptography – PKC 2019*, pages 97–127, Cham, 2019. Springer International Publishing.

Supplementary material: Deferred Proofs

A Proofs in Section 2

A.1 Proof of Proposition 2.1

Let $\mathbf{S}^{(L)} = \mathsf{sk} \in \mathbb{Z}_q$ be a secret key that we want to share and $\mathbf{U}^{(L)} = \mathbf{D}$ be a share matrix. Let $\mathbf{R}^{(L)} = (\mathbf{S}^{(L)}, \mathbf{r}^{(L)})^T \in \mathbb{Z}_q^m$ be a vector where $\mathbf{r}^{(L)} = (r_2^{(L)}, \cdots, r_m^{(L)})$ is a random vector and its elements are uniformly sampled, $r_2^{(L)}, \cdots, r_m^{(L)} \leftarrow \mathbb{Z}_q$. By repeated matrix multiplications, we can generalize that for an integer $1 \leq i \leq L$, level-$(L-i)$ secret shares $\mathbf{S}^{(L-i)}$, share matrix $\mathbf{U}^{(L-i)}$, and input vector $\mathbf{R}^{(L-i)}$ are as follows:

$$
\begin{aligned}
\mathbf{S}^{(L-i)} &= \mathbf{U}^{(L-i+1)} \cdot \mathbf{R}^{(L-i+1)} \in \mathbb{Z}_q^{\ell^i}, \\
\mathbf{U}^{(L-i)} &= \mathbf{I}_{\ell^i} \otimes \mathbf{D} = \begin{bmatrix} \mathbf{D} & 0 & \cdots & 0 \\ 0 & \mathbf{D} & \cdots & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{D} \end{bmatrix} \in \mathbb{Z}_q^{\ell^{i+1} \times m \cdot \ell^i}, \\
\mathbf{R}^{(L-i)} &= (\mathbf{S}^{(L-i)}[1], r_2^{(L-i)}, \cdots, r_m^{(L-i)}, \mathbf{S}^{(L-i)}[2], r_{m+2}^{(L-i)} \cdots, r_{m\cdot\ell^i}^{(L-i)})^T \in \mathbb{Z}_q^{m\cdot\ell^i} \\
&= \mathbf{P}^{(L-i)} \cdot (\mathbf{S}^{(L-i)}[1], \mathbf{S}^{(L-i)}[2], \cdots, \mathbf{S}^{(L-i)}[\ell^i], r_2^{(L-i)}, \cdots, r_{m\cdot\ell^i}^{(L-i)})^T \\
&= \mathbf{P}^{(L-i)} \cdot (\mathbf{S}^{(L-i)}, \mathbf{r}^{(L-i)})^T,
\end{aligned}
$$

where $\mathbf{P}^{(L-i)}$ is a permutation matrix and $r_2^{(L-i)}, \cdots, r_{m\cdot\ell^i}^{(L-i)} \leftarrow \mathbb{Z}_q$ and $\mathbf{r}^{(L-i)} = (r_2^{(L-i)}, \cdots, r_{m\cdot\ell^i}^{(L-i)}) \in \mathbb{Z}_q^{(m-1)\cdot\ell^i}$.

Actually, the permutation matrix $\mathbf{P}^{(L-i)}$ is defined to simply express the share matrix of $\mathsf{SS}^{(L)}$. Now, we can compute the share matrix $\mathbf{M}^{(L)}$ such that $\mathbf{S}^{(0)} = \mathbf{M}^{(L)} \cdot (\mathsf{sk}, \mathbf{r}^{(L)}, \mathbf{r}^{(L-1)}, \cdots, \mathbf{r}^{(1)})$. Then we can express $\mathbf{S}^{(0)}$ as follows.

$$
\begin{aligned}
\mathbf{S}^{(0)} &= \mathbf{U}^{(1)} \cdot \mathbf{R}^{(1)} = \mathbf{U}^{(1)} \cdot \mathbf{P}^{(1)} \cdot \begin{bmatrix} \mathbf{S}^{(1)} \\ \mathbf{r}^{(1)} \end{bmatrix} \\
&= \mathbf{U}^{(1)} \cdot \mathbf{P}^{(1)} \cdot \begin{bmatrix} \mathbf{U}^{(2)} \cdot \mathbf{P}^{(2)} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{S}^{(2)} \\ \mathbf{r}^{(2)} \\ \mathbf{r}^{(1)} \end{bmatrix}
\end{aligned}
$$

$$= \mathbf{U}^{(1)} \cdot \mathbf{P}^{(1)} \cdots \begin{bmatrix} \mathbf{U}^{(L)} \cdot \mathbf{P}^{(L)} & \mathbf{0} \cdots \mathbf{0} \\ \mathbf{0} & \mathbf{I} \cdots \mathbf{0} \\ \vdots & \vdots \ddots \vdots \\ \mathbf{0} & \mathbf{0} \cdots \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{S}^{(L)} \\ \mathbf{r}^{(L)} \\ \vdots \\ \mathbf{r}^{(1)} \end{bmatrix}$$

$$\mathbf{M}^{(L)} = \mathbf{U}^{(1)} \cdot \mathbf{P}^{(1)} \cdot \begin{bmatrix} \mathbf{U}^{(2)} \cdot \mathbf{P}^{(2)} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \cdot \cdots \cdot \begin{bmatrix} \mathbf{U}^{(L)} \cdot \mathbf{P}^{(L)} & \mathbf{0} \cdots \mathbf{0} \\ \mathbf{0} & \mathbf{I} \cdots \mathbf{0} \\ \vdots & \vdots \ddots \vdots \\ \mathbf{0} & \mathbf{0} \cdots \mathbf{I} \end{bmatrix},$$

where $\mathbf{U}^{(L)} \cdot \mathbf{P}^{(L)} = \mathbf{D}$. Then, we show that $\mathsf{SS}^{(L)}$ is a linear secret sharing scheme of sk for all positive integer L. □

A.2 Lemma for $(2s-1)$-input majority function

Lemma A.1 ([28]) *Let $X_1, \cdots, X_{2s-1} \leftarrow \{0,1\}$ be three independent identically distributed random variables, and $p := \Pr[X_i = 1]$ for all i. Then, following properties hold:*

1. $p' := \Pr[\mathit{MAJ}_s(X_1, \cdots, X_{2s-1}) = 1] = \sum_{j=0}^{s-1} \binom{2s-1}{j} \cdot p^{2s-1-j}(1-p)^j.$
2. $\delta := p - 0.5$, *it holds that* $p' = 0.5 + \left(\frac{2s-1}{4^{s-1}} \cdot \binom{2s-2}{s-1} - O(\delta) \right) \cdot \delta.$
3. $p' < \binom{2s-1}{s} p^s.$

Proof. 1. $\Pr[\mathsf{MAJ}_s(X_1, \cdots, X_{2s-1}) = 1] = \Pr[X_1 + \cdots + X_{2s-1} = s, \cdots, 2s-1] = \sum_{j=0}^{s-1} \binom{2s-1}{j} \cdot p^{2s-1-j}(1-p)^j.$

2. Since $p = 0.5 + \delta$,

$$\begin{aligned} p' &= \sum_{j=0}^{s-1} \binom{2s-1}{j} \cdot \left(\frac{1}{2} + \delta\right)^{2s-1-j} \left(\frac{1}{2} - \delta\right)^j \\ &= \sum_{j=0}^{s-1} \binom{2s-1}{j} \cdot \left(\frac{1}{2^{2s-1}} + (2s-1-j) \cdot \frac{1}{2^{2s-2}} \cdot \delta - j \cdot \frac{1}{2^{2s-2}} \cdot \delta + O(\delta^2) \right) \\ &= \frac{1}{2} + \sum_{j=0}^{s-1} \frac{2s-1}{2^{2s-2}} \cdot \left(\binom{2s-2}{j} \cdot \delta - \binom{2s-2}{j-1} \cdot \frac{1}{2^{2s-2}} \cdot \delta \right) + O(\delta^2) \\ &= \frac{1}{2} + \left(\frac{2s-1}{4^{s-1}} \cdot \binom{2s-2}{s-1} - O(\delta) \right) \cdot \delta. \end{aligned}$$

3.

$$p' = \sum_{j=0}^{s-1} \binom{2s-1}{j} \cdot p^{2s-1-j}(1-p)^j$$

$$
\begin{aligned}
&< \binom{2s-1}{s-1} p^s \cdot \sum_{j=0}^{s-1} \binom{s-1}{j} \cdot p^{s-1-j}(1-p)^j \\
&= \binom{2s-1}{s-1} p^s.
\end{aligned}
$$

By Lemma A.1, the iteration number i_s to construct an N-input majority function is $\log_{c_s} N + O(1) + \log_s N$ where $c_s = \frac{2s-1}{4^{s-1}} \cdot \binom{2s-2}{s-1}$. Then, the total number of secret shares is $O((2s-1)^{i_s}) \approx O(N^{\log_{c_s}(2s-1)+\log_s(2s-1)})$.

In [15], the authors provide lower and upper bound of the size of c_s.

$$
\frac{1}{\sqrt{\pi}} \cdot \frac{2s-1}{\sqrt{s-1/2}} < \frac{2s-1}{4^{s-1}} \cdot \binom{2s-2}{s-1} < \frac{1}{\sqrt{\pi}} \cdot \frac{2s-1}{\sqrt{s-1}}
$$

Therefore, $\log_{c_s}(2s-1) + \log_s(2s-1)$ is a decreasing function for s and lower bound is 3. $\square$

A.3 Proof of Lemma 2.3

Proof. For $j \in [N]$ and $S \subset [N] \cup \{0\}$ with the threshold value t, the Lagrange coefficient $\lambda_{i,j}^S$ can be represented by $\prod_{m \in S \setminus \{i\}} \frac{j-m}{i-m}$ for all $i \in S$. Then, the numerator and denominator of Lagrange coefficient $\lambda_{i,j}^S$ have the following properties:

$$
\left(\prod_{m \in S\setminus\{i\}} (j-m)\right) \Bigg| \left(\prod_{m \in N\setminus\{j\}} (j-m)\right) = (-1)^{N-j} j! \cdot (N-j)! \Bigg| N!,
$$
$$
\left(\prod_{m \in S\setminus\{i\}} (i-m)\right) \Bigg| \left(\prod_{m \in N\setminus\{i\}} (i-m)\right) = (-1)^{N-i} i! \cdot (N-i)! \Bigg| N!.
$$

Therefore, $N! \cdot \lambda_{i,j}^S$, and $N! \cdot \frac{1}{\lambda_{i,j}^S}$ are both integers and their bound are $(N!)^2$.

B Proofs in Section 5

The proofs are almost the same as the original proofs in the full version of [9].

B.1 Proof of Theorem 5.7

It is obvious that the encryption (evaluation) of TFHE is equal to the encryption of FHE. Thus, the compactness of TFHE automatically holds whenever FHE satisfies the compactness. $\square$

B.2 Proof of Theorem 5.8

By Construction 5.6, the following satisfies:

- Given the secret key of fully homomorphic encryption fhesk, it is splitted as follows:
$$(\mathsf{share}_1^{(0)}, \cdots, \mathsf{share}_{(2s-1)^L}^{(0)}) \leftarrow \mathsf{SS.Share}(\mathsf{fhesk}, \mathbb{A}_t)$$
where SS a level-L Shamir secret sharing scheme with $\mathbf{V}_s$ in Section 4.2.
- The setup algorithm returns $\mathsf{pk} = \mathsf{fhepk}$ and $\mathsf{sk}_i = \{\mathsf{share}_j^{(0)}\}_{j \in T_i}$ for $i \in [N]$.
- The partial decryption algorithm outputs $p_i = \{\hat{\mathbf{p}}_j^{(0)}\}_{j \in T_i}$, where
$$\hat{\mathbf{p}}_j^{(0)} = \mathsf{FHE.Dec}_0(\mathsf{share}_j^{(0)}, \mathsf{ct}) + ((2s-1)!)^L \cdot e_j \in \mathbb{Z}_q$$
for every $j \in T_i = \{j \mid P_i \textit{ has } \mathsf{share}_j^{(0)}\}$ and any (valid) ciphertext ct.

Let $T^{(0)} \subseteq \cup_{i \in S} T_i$ be the minimal valid share set for $S \in \mathbb{A}_t$. Then, $\{\mathsf{share}_j^{(0)}\}_{j \in T^{(0)}}$ can be recovered to the secret key sk using the iterative secret sharing on $\mathbf{V}_s$ in Section 4.2.

Let $T^{(i)}$ be a family of indices such that $T^{(i)} = \{k \mid \mathsf{share}_k^{(i)}$ can be recovered by $\{\mathsf{share}_j^{(0)}\}_{j \in T^{(0)}}\}$ for $1 \leq i \leq L$. Then, the correctness of TreeSSS, fhesk can be expressed as follows:

$$\begin{aligned}
\mathsf{fhesk} &= \mathsf{share}^{(L)} \\
&= \sum_{j_{L-1} \in T^{(L-1)}} \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \mathsf{share}_{j_{L-1}}^{(L-1)} \\
&= \sum_{j_{L-2} \in T^{(L-2)}} \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \lambda_{j_{L-2}}^{S_{j_{L-2}}^{(L-2)}} \cdot \mathsf{share}_{j_{L-2}}^{(L-2)} \\
&\;\vdots \\
&= \sum_{j \in T^{(0)}} \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdots \lambda_{j_1}^{S_{j_1}^{(1)}} \cdot \lambda_j^{S_j^{(0)}} \cdot \mathsf{share}_j^{(0)}
\end{aligned}$$

where j_k is an index of level-k secret shares which uses $\mathsf{share}_j^{(0)}$ to recover itself, $S_{j_k}^{(k)}$ is a set of level-k secret shares including $\mathsf{share}_{j_k}^{(k)}$ which recover a level-$(k+1)$ secret share $\mathsf{share}_{j_{k+1}}^{(k+1)}$, and the Lagrange coefficient $\lambda_{j_k}^{S_{j_k}^{(k)}}$ are obtained from the Lagrange polynomial of Shamir's secret sharing $\mathsf{SS}^{(1)}$.

On top of this construction, the linearity of $\mathsf{FHE.Dec}_0$ provides the following relation:

$$\sum_{j \in T^{(0)}} \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdots \lambda_{j_1}^{S_{j_1}^{(1)}} \cdot \lambda_j^{S_j^{(0)}} \cdot \hat{\mathbf{p}}_j^{(0)}$$

$$= \sum_{j\in T^{(0)}} \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdots \lambda_{j_1}^{S_{j_1}^{(1)}} \cdot \lambda_{j}^{S_{j}^{(0)}} \cdot (\mathsf{FHE.Dec}_0(\mathsf{share}_j^{(0)}, \mathsf{ct}) + ((2s-1)!)^L e_j)$$

$$= \mathsf{FHE.Dec}_0\left(\sum_{j\in T^{(0)}} \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdots \lambda_{j_1}^{S_{j_1}^{(1)}} \cdot \lambda_{j}^{S_{j}^{(0)}} \cdot \mathsf{share}_j^{(0)}, \mathsf{ct}\right)$$

$$+ \sum_{j\in T^{(0)}} \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdots \lambda_{j_1}^{S_{j_1}^{(1)}} \cdot \lambda_{j}^{S_{j}^{(0)}} \cdot ((2s-1)!)^L e_j$$

$$= \mathsf{FHE.Dec}_0(\mathsf{FHEsk}, \mathsf{ct}) + \sum_{j\in T^{(0)}} \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdots \lambda_{j_1}^{S_{j_1}^{(1)}} \cdot \lambda_{j}^{S_{j}^{(0)}} \cdot ((2s-1)!)^L e_j$$

$$= \mu\lfloor\frac{q}{2}\rceil + e + \sum_{j\in T^{(0)}} \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdots \lambda_{j_1}^{S_{j_1}^{(1)}} \cdot \lambda_{j}^{S_{j}^{(0)}} \cdot ((2s-1)!)^L e_j.$$

Consequently, $\mathsf{FHE.Dec}_1(\sum_{j\in T^{(0)}} \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdots \lambda_{j_1}^{S_{j_1}^{(1)}} \cdot \lambda_{j}^{S_{j}^{(0)}} \cdot \hat{\mathbf{p}}_j^{(0)})$ returns the correct messages when the error term is appropriately bounded because of Definition 3.12.

Let e_{sm} be a noise smudging error of the form $\sum_{j\in T^{(0)}} \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdots \lambda_{j_1}^{S_{j_1}^{(1)}} \cdot \lambda_{j}^{S_{j}^{(0)}} \cdot ((2s-1)!)^L e_j$. By Lemma 2.3, it holds that $|e_{sm}| \leq ((2s-1)!)^{2L} \cdot (2s-1)^L \cdot B_{sm}$, and it implies $|e + e_{sm}| \leq B + ((2s-1)!)^{2L} \cdot (2s-1)^L \cdot B_{sm} \leq \lfloor \frac{q}{4} \rceil$. Thus, FHE satisfies the its correctness, which directly implies that TFHE also satisfies the correctness. $\square$

B.3 Proof of Theorem 5.9

The encryption in TFHE is equivalent to that in FHE. As per the privacy of secret sharing, if a set of partial secret shares $\{\mathsf{sk}_i\}_{i\in S}$ are kept confidential, then they do not reveal any information about the secret key sk when $S \notin \mathbb{A}_t$. This means that the security of FHE implies the semantic security of TFHE.