Improved Universal Thresholdizer from Iterative Shamir Secret Sharing

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, introduced at CRYPTO'18, is a cryptographic scheme that transforms any cryptosystem into a threshold variant, thereby enhancing its applicability in threshold cryptography. It enables black-box construction of one-round threshold signature schemes based on the Learning with Errors problem, and similarly, facilitates one-round threshold ciphertext-attack secure public key encryption when integrated with non-threshold schemes.

Current constructions of universal thresholdizer are fundamentally built upon linear secret sharing schemes. One approach employs Shamir's secret sharing, which lacks compactness and results in ciphertext sizes of $O(N \log N)$, and another approach uses $\{0,1\}$-linear secret sharing scheme ($\{0,1\}$-LSSS), which is compact but induces high communication costs due to requiring $O(N^{5.3})$ secret shares.

In this work, we introduce a communication-efficient universal thresholdizer by revising the linear secret sharing scheme. We propose a specialized linear secret sharing scheme, called TreeSSS, which reduces the number of required secret shares $O(N^{3+2.3/\log\log N})$ while maintaining the compactness of the universal thresholdizer.

TreeSSS can also serve as a subroutine for constructing lattice based t-out-of-N threshold cryptographic primitives such as threshold fully homomorphic encryptions and threshold signatures. In this context, TreeSSS offers the advantage of lower communication overhead due to the reduced number of secret shares involved.

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

1 Introduction

A t-out-of-N threshold cryptography [25, 26, 28] 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) [14] 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) [39, 47], and a non-interactive commitment scheme [11]. 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 [34,51], which employs binary coefficients to recover the secret from the distributed secret shares 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 shares, which can result in substantial communication overhead with a cost of $O(N^{5.3} \log N)$.

1.1 This work

Our main contribution is to propose a communication-efficient UT by constructing an efficient linear secret sharing scheme for t-out-of-N threshold access structure, called TreeSSS. This linear secret sharing scheme reduces the number of shared keys compared to the standard $\{0,1\}$-LSSS, leading to less secret shares 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 a compact UT. The reduced communication overhead makes it a promising candidate for various applications. These include but are

[4] [14, 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.

not limited to threshold signatures [2], threshold multi-key FHE schemes [5], and decentralized Attribute-Based Encryption (ABE) [52].

TreeSSS is devised through the iterative application of s-out-of-$2s-1$ Shamir's secret sharing scheme, where $s \ll N$. Its results exactly match the *optimal amplification* employed for constructing the large input majority function using small input majority functions [36]. The results are given in Table 1.

We note that the iterative construction is specially applicable to a t-out-of-N threshold access structure. Furthermore, only formulas consisting of AND/OR gates are known to be transformed into linear secret sharing schemes [14]. Notably, methods that employ majority gates have been considered relatively inefficient for constructing linear secret sharing schemes [34].

Last, we point out an overlooked probability issue that was not fully considered when constructing the share matrix $\mathbf{M}$ using $\{0,1\}$-LSSS. This issue originates from Valiant's monotone Boolean formula in [51], which serves as the foundation structure for $\{0,1\}$-LSSS. According to Valiant's formula Lemma A.3, it guarantees that a generated circuit will corresponds to a majority circuit with at least 1/2 probability. This implies that a matrix $\mathbf{M}$, generated through $\{0,1\}$-LSSS, has at least 1/2 probability of being a (well-constructed) share matrix as intended.

To address this, we adjust the construction, allowing for $\mathbf{M}$ to be obtained with overwhelming probability while preserving the asymptotically same the number of secret shares. Further details and modifications are discussed in Section 2.3.

Secret Sharing Scheme		Structure	$O(\log q)$	# of keys
Previous	Shamir SS	t-out-of-N	$O(N \log N)$	N
	$\{0,1\}$-LSSS[6]		$O(\log N)$	$O(N^{5.3})$
TreeSSS	$\mathsf{SS}(3,2)$	t-out-of-N	$O(\log N)$	$O(N^{4.3})$
	$\mathsf{SS}(2s-1,s)$		$O(s \log s \cdot \log N)$	$O(N^{3+\frac{2.3}{\log s}})$
	$\mathsf{SS}(\log(N^2/2), \log N)$		$O((\log N)^2 \cdot \log\log N)$	$O(N^{3+\frac{2.3}{\log\log N}})$

Table 1: The comparison results between the previous TFHE and ours. The column 'structure' indicates the access structure of secret sharing schemes. The remaining columns carry the same meaning as those in the preceding table. $\mathsf{SS}(N,t)$ indicates Shamir's secret sharing scheme for t-out-of-N threshold. The final row, which corresponds to $\mathsf{SS}(2s-1,s)$, represents the asymptotic behavior as N tends to infinity, with s being constant and N being sufficiently large. Communication cost is the product of the size of the ciphertext and the total number of keys.

[6] [14,38] propose the definition of $\{0,1\}$-LSSS for arbitrary access structure, but they only instantiated $\{0,1\}$-LSSS for t-out-of-N.

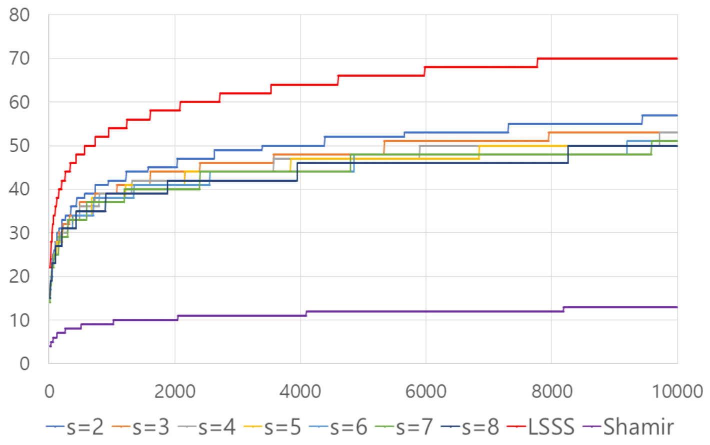
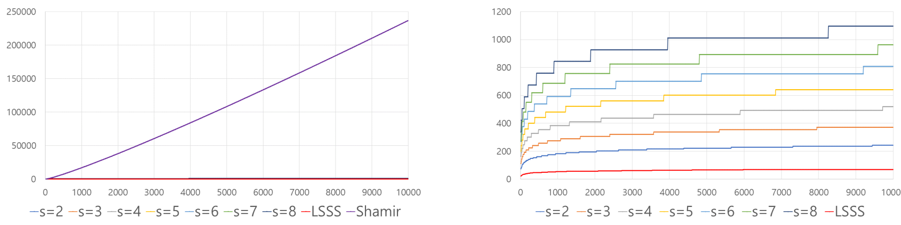


Fig. 1: Comparison the number of secret shares: Ours, $\{0,1\}$-LSSS, and Shamir secret sharing. Log-size of the number of secret shares according to number of parties N.



(a) Log-size of decryption error bound according to number of parties N. The purple line indicates Shamir's secret sharing scheme.

(b) (Zoom in) Log-size of decryption error bound except for Shamir's secret sharing scheme.

Fig. 2: Comparison decryption error bound: Ours, $\{0,1\}$-LSSS, and Shamir secret sharing

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 [6,15,23,40]. Our focus is on directly improving $\{0,1\}$-LSSS, while [15] uses it as is. We thus believe our TreeSSS and the technique in [15] can be combined to create an efficient

TFHE and UT construction. We further note that [40] improved the bootstrapping technique of FHEW/TFHE to achieve threshold FHE, while [15, 23] both achieved a polynomial modulus-to-noise ratio TFHE from LWE using Rènyi divergence and the noise flooding technique. [23] was specialized for Torus-FHE [22], while [15] can be built from any FHE scheme. These schemes are built from $\{0,1\}$-LSSS, and require approximately $N^t \approx \binom{N}{t}$ operations during the setup protocol. This adjustment is intended to boost practical performance, particularly when working with small N and t. In contrast, our algorithm focuses on achieving asymptotic improvements for arbitrary N and t. Therefore, these schemes are not within the scope of this paper. Nevertheless, in Figure 1, we gave a graphical comparison of number of secret shares for ours, $\{0,1\}$-LSSS, and Shamir secret sharing to indirectly support the superiority of TreeSSS for arbitrary N and t.

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 [34,36]. However, these constructions are not useful for threshold circuits similar to TFHE. For example, [34] 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 [51]. A paper [6] introduces specialized fields with a characteristic of 2, focusing specifically on the threshold structures of 2-out-of-N and $(N-1)$-out-of-N. [37] improved upon the result in [51], which had a size of $O(n^{1+\sqrt{2}})$. However, [15] argued that it is unclear how to construct TFHE using the improved result in [37] because of their circuit construction.

[36] found that the optimal formula for general majority can be expressed with the $(2s-1)$-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 [12, 13, 19, 24, 30–33, 42, 44, 50], but most of them are based on prequantum objects like ECDSA. There have only been two round-optimal threshold signature schemes from lattices [2, 14]. The first one was built from UT via compact TFHE [14], 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 [4, 18, 35, 43, 45, 46]. 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 [45] and on-the-fly MPC property [43].

Ramp secret sharing. Ramp secret sharing was first introduced in [10]. This scheme differentiates between the number of *reconstruction parties*, τ_c, who can recover the secret, and the number of *privacy parties*, τ_p, who gain no information about the secret. As such, ramp secret sharing serves as a more efficient, albeit weaker, variant of traditional secret sharing schemes. Recently, there are several studies using ramp setting such as weighted threshold cryptosystems [8, 29] and blackbox near-threshold secret sharing [3]. However, it's crucial to acknowledge that the ramp setting may not be applicable to all practical scenarios. This limitation arises from the *'gray area'* where the number of participating parties falls within the interval (τ_c, τ_p). In such cases, neither correctness nor reconstruction can be guaranteed by the scheme.

2 Technical Overview

This section provides a technical overview of a new secret sharing scheme, called TreeSSS. We further provide how to construct TFHE from TreeSSS.

2.1 Current State of Threshold Fully Encryption Scheme from Secret Sharing Scheme

We briefly introduce a core technique TFHE from current secret sharing schemes, Shamir's secret sharing scheme [48] and $\{0,1\}$-LSSS [14, 38], to describe the improvement of new TFHE and UT induced by TreeSSS.

Assume a LWE based fully homomorphic encryption scheme FHE such as [16, 17, 20, 27] is given. Let ct be a ciphertext of a message $m \in \{0,1\}$ and $\mathsf{sk} \in \mathbb{Z}_q^n$ be a secret key of FHE with respect to LWE parameters n and q. The decryption algorithm of FHE takes ct and sk as input and returns a message via computing an inner product $\langle \mathsf{ct}, \mathsf{sk} \rangle \in \mathbb{Z}_q$ and returns a message m after some modifications of the inner product.

Suppose that sk is distributed into shares $\mathsf{sk}_i \in \mathbb{Z}_q^n$ among parties. The shares satisfy $\mathsf{sk} = \sum_i c_i \cdot \mathsf{sk}_i$ for some coefficient $c_i \in \mathbb{Z}_q$. Then, the decryption process of FHE can be interpreted as follows:

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

The linear secret sharing scheme enables us to compute a pair (c_i, sk_i) and securely share the secret share sk_i to each party. Thus, through linear secret sharing scheme, one can construct t-out-of-N *threshold* FHE. Unfortunately, it leaks the information of the secret share sk_i from $\langle \mathsf{ct}, \mathsf{sk}_i \rangle$. To avoid this leakage, the decryptor injects small noises e_i to $\langle \mathsf{ct}, \mathsf{sk}_i \rangle$, so the decryption process is performed as follows:

$$\sum_i c_i(\langle \mathsf{ct}, \mathsf{sk}_i \rangle + e_i) \bmod q = \langle \mathsf{ct}, \mathsf{sk} \rangle + \sum_i c_i \cdot e_i \bmod q.$$

It is easy to confirm that $\sum_i c_i \cdot e_i$ should be small for the correctness.

Under this circumstance, Shamir's secret sharing scheme [48] generates a pair (c_i, sk_i) such that the recovery coefficient c_i is the Lagrange coefficient $\lambda_i \in \mathbb{Q}$.

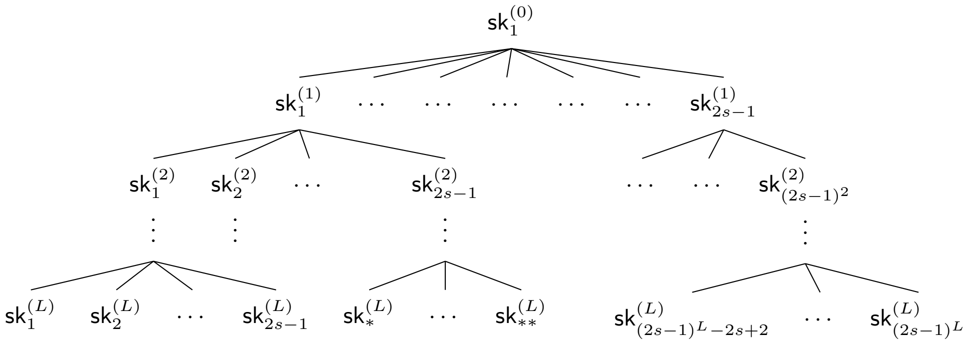
Thus, in order to incorporate Shamir's secret sharing scheme into FHE, one must multiply $N!$ by λ_i to ensure that $c_i \cdot e_i = \lambda_i \cdot e_i$ lies over $\mathbb{Z}_q$. This implies that q should be at least bigger than $N! \cdot \lambda_i$. Consequently, $\log q$ should be set to $O(N \log N)$.

On the other hand, $\{0,1\}$-LSSS [14,38] satisfies that the recovery coefficient c_i is binary, so it achieves the compactness of threshold FHE. That is, $\log q$ sets to $O(\log N)$. However, the $\{0,1\}$-LSSS requires a substantial number of secret shares. In fact, for the correctness and privacy of the $\{0,1\}$-LSSS, $O(N^{5.3})$ secret shares are distributed. This is substantial compared to Shamir's secret sharing scheme, which shares only N secret shares.

2.2 TreeSSS: Tree Secret Sharing Scheme for t-out-of-N Threshold Access Structure

This section primarily provides an overview of generating a tree secret sharing scheme for the $\frac{N+1}{2}$-out-of-N threshold access structure, where N is an odd number. Subsequently, we expand this concept to develop a tree secret sharing scheme for an arbitrary threshold access structure (see Section 4.3).

Firstly, we observe that the $\frac{N+1}{2}$-out-of-N threshold circuit can be interpreted as an N-input majority function. Consequently, the formulation of a secret sharing scheme for the $\frac{N+1}{2}$-out-of-N threshold access structure could conceivably correspond to the construction of an N-input majority function.



Distribute level-L secret shares $\{\mathsf{sk}_i^{(L)}\}$ to each party [7]

Fig. 3: High-level Overview of TreeSSS

We observe that circuit compositions to generate N-input majority function from $(2s-1)$-input majority function, where $s \ll N$ corresponds to repeatedly

[7] The method for distributing level-L secret shares is exactly the same as $\{0,1\}$-LSSS. Informally, the distributor randomly partitions the set $1,\ldots,3^L$ into N subsets, and sends the level-L secret shares corresponding to the indices within these N subsets to each respective party We leave the detailed method in Section 4.2.

using Shamir's secret sharing for s-out-of-$(2s-1)$ threshold structure as in Fig. 3. This observation is in strict agreement with the preceding results for constructing an N-input majority function proposed by Gupta and Mahajan [36]. They proved that the $(2s-1)$-input majority function can be repeatedly used to construct N-input majority functions, even when $s \ll N$.

This approach generalizes $\{0,1\}$-LSSS in terms of the circuit representation of threshold circuits. This is because $\{0,1\}$-LSSS can be also interpreted as applying the iterative technique on specific circuits from the monotone Boolean formula for threshold circuits proposed by Valiant [51].[8] To demonstrate this concept, we present the following lemmas, which outlines the construction of majority functions from a 3-input majority function.

The first lemma asserts that the utilization of small input majority functions can increase the probability of each inputs.

Lemma 2.1 (Adaptation from [34, 36]) *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. $1 - p' < 3(1-p)^2$.

Intuitively, Lemma 2.1 says that we can increase the bias $\delta = p - 0.5$ through iterations of MAJ_3: it holds that $\delta' = p' - 0.5 \geq (1.5 - 2\delta^2) \cdot \delta \geq \delta$ for any $\delta < 0.5$. That is, the iteration technique yields the probability amplification. The proof is deferred to Suppl. B.1.

Under the fact that majority circuits MAJ_3 correspond to 2-out-of-3 Shamir's secret sharing scheme $\mathsf{SS}(3,2)$, this lemma inspires the following design of TreeSSS: Let sk be a secret key that we want to share. For the leveled secret sharing, we regard sk as level-0 secret share $\mathsf{sk}_1^{(0)}$. Suppose that level-i secret shares $\{\mathsf{sk}_j^{(i)}\}_{j \in [3^i]}$ are given. For each $0 \leq i < L$, by applying $\mathsf{SS}(3,2)$ to each $\mathsf{sk}_j^{(i)}$, we obtain level-$(i+1)$ secret shares of the form $\{\mathsf{sk}_j^{(i+1)}\}_{j \in [3^{i+1}]}$ for priory defined parameter L. More precisely, for every i, j, $\mathsf{SS}(3,2)$ can split $\mathsf{sk}_j^{(i)}$ into $\{\mathsf{sk}_{3j-2}^{(i+1)}, \mathsf{sk}_{3j-1}^{(i+1)}, \mathsf{sk}_{3j}^{(i+1)}\}$. By iteratively using $\mathsf{SS}(3,2)$, we obtain level-L secret shares $\{\mathsf{sk}_j^{(L)}\}_{j \in [3^L]}$. Last, we distribute level-L secret shares $\{\mathsf{sk}_j^{(L)}\}_{j \in [3^L]}$ to N parties $P = \{P_1, \ldots, P_N\}$.

As the perspective of secret sharing schemes, Lemma 2.1 reinterprets the probability amplification of success probability of recovering sk from secret shares $\{\mathsf{sk}_j^{(L)}\}$: Given any $S \subset P$, we define a random variable $X_{i,j}$ as follows:

$$X_{i,j}(S) = \begin{cases} 1 & \text{if } S \text{ can recover the secret share } \mathsf{sk}_j^{(i)} \\ 0 & \text{otherwise} \end{cases}$$

[8] In the supplementary material, we provide how to interpret $\{0,1\}$-LSSS as iteration of small circuits. Goldreich [34] proved that representing threshold circuits by monotone Boolean formula yields inefficient TFHE construction in terms of the number of secret shares to each party. Indeed, it requires $O(N^7)$ secret shares to construct (compact) TFHE.

for every $0 \leq i \leq L$ and $j \in [3^i]$. Note that for every j, it holds that $\Pr[X_{L,j} = 1] = \Pr[S \text{ has a secret share } \mathsf{sk}_j^{(L)}] = \frac{|S|}{N}$.[9] Furthermore, it also satisfies that if $i < L$, then $\mathsf{sk}_j^{(i)}$ can be recovered whenever S can gather two or three level-$(i+1)$ secret shares in $\{\mathsf{sk}_{3j-2}^{(i+1)}, \mathsf{sk}_{3j-1}^{(i+1}, \mathsf{sk}_{3j}^{(i+1)}\}$. Hence, it holds that

$$\Pr[X_{i,j} = 1] = \Pr[\mathsf{MAJ}_3(X_{i+1,3j-2}, X_{i+1,3j-1}, X_{i+1,3j}) = 1].$$

Lemma 2.1 posits that constructing 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. The following lemma describes the level required to construct a N-input majority function with $(2s-1)$-input majority function.

Lemma 2.2 (Adaptaion from [36]) *Given an odd number N of parties, one can construct the majority function MAJ_N from small majority gates MAJ_{2s-1} with at least 1/2 success probability. This construction requires iteratively applying MAJ_{2s-1}-gates up to level $\log_{c_s}(N) + \log_s N + O(1)$, where $c_s = \frac{2s-1}{2^{2s-2}} \cdot \binom{2s-2}{s-1}$.*

Lemma 2.2 indicates that the construction of N-input majority functions is possible through the use of a 3-input majority function with a size of $O(N^{4.29})$ or through a $(2s-1)$-input majority function with a size of $O(N^{3+O(1/\log s)})$. Indeed, we correspond a $(2s-1)$-input majority circuit to $\mathsf{SS}(2s-1, s)$, where $\mathsf{SS}(2s-1, s)$ is Shamir's secret sharing scheme for s-out-of-$(2s-1)$ threshold access structure. Fig. 3 indicates the high-level overview of TreeSSS. Working backwards from the leaf nodes of the tree to root, we can combine the secret sk. The detailed construction of TreeSSS will be presented in Section 4.2.

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 $t > \frac{N+1}{2}$, we start by constructing a TreeSSS for t-out-of-$(2t-1)$. If $2t - 1 > N$, we simply disregard the extra secret shares. Similarly, if $t < \frac{N+1}{2}$, 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.

2.3 Previous Flaw: Concealed probability in $\{0,1\}$-LSSS

The $\{0,1\}$-LSSS described in [14, 38] depends on the probabilistic construction of monotone Boolean formulas. The core lemmas Lemma A.3 (for $\{0,1\}$-LSSS) and Lemma 2.2 argue that that the probability of constructing the circuits is at least 1/2.

[9] Assume that a dealer distributes level-L secret shares to each party P_i for $i \in [N]$. Then, the probability of each party P_i having $\mathsf{sk}_j^{(L)}$ is equal to $\frac{1}{N}$ for all i, j. It means that the probability of a subset S having $\mathsf{sk}_j^{(L)}$ is $\sum_{P_i \in S} \frac{1}{N} = \frac{|S|}{N}$.

This only ensures that we can construct $\{0,1\}$-LSSS with $1/2$ probability depending on the distribution of secret shares. Therefore, a dealer should redistribute the secret shares to the parties until constructing a proper secret sharing scheme. Consequently, for usage of these schemes, it is necessary to verify that the share algorithm functions as intended. Thus, it is necessary to verify that the secret sharing scheme actually satisfies both correctness and privacy. However, the time complexity for this verification is exponential in N rendering it impractical for large N. This verification step is present in our TreeSSS because our scheme also relies on the construction of probabilistic circuits Lemma 2.2, which yields the additional time costs.

The success probability (at least $1/2$) of Lemma 2.2 is derived from Lemma 2.1 and Lemma 2.3.

Lemma 2.3 (**[51]**) *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 $\Pr[F \equiv \mathit{MAJ}_N] \geq 1/2$.

To mitigate the extra computational overhead, we make a minor adjustment to Lemma 2.3 and the iteration level L, it leads to increase success probability of a threshold structure. More precisely, we add an additional parameter κ. If the failure probability of each cases is less than $2^{-N-\kappa-1}$, a suitable formed share matrix $\mathbf{M}$ can be acquired with a probability of $1 - \frac{1}{2^\kappa}$ with an additional levels $\log_s \kappa$.

Secret Sharing Scheme		Structure	$O(\log q)$	# of keys
Previous	Shamir SS	t-out-of-N	$O(N \log N)$	N
	$\{0,1\}$-LSSS		$O(\log N)$	$O(N^{3.3} \cdot (N+\kappa)^2)$
TreeSSS	$\mathsf{SS}(3,2)$	t-out-of-N	$O(\log N)$	$O(N^{2.72} \cdot (N+\kappa)^{1.58})$
	$\mathsf{SS}(19,10)$		$O(\log N)$	$O(N^{2.34} \cdot (N+\kappa)^{1.28})$
	$\mathsf{SS}(99,50)$		$O(\log N)$	$O(N^{2.22} \cdot (N+\kappa)^{1.17})$
	$\mathsf{SS}(2s-1,s)$		$O(s \log s \cdot \log N)$	$O(N^{2+\varepsilon} \cdot (N+\kappa)^{1+\varepsilon})$[10]

Table 2: The comparison results between the previous TFHE and ours after the modification.

Consequently, the number of shares can be expressed as $O(N^{\log_{c_s}(2s-1)} \cdot (N+\kappa)^{\log_s(2s-1))})$, which is asymptotically equivalent to the number of shares

[10] $\varepsilon = O(1/\log s)$.

in Theorem 4.6. A comprehensive proof of the modification of L can be found in Suppl. C. We also note that in case of $\{0,1\}$-LSSS, $O(N^{5.3})$ is changed into $O(N^{3.3} \cdot (N+\kappa)^2)$. For detailed impacts of this modification, please refer to Table 2.

Certainly, for the sake of clearer exposition, we choose not to use κ throughout this paper, even though it has probabilistic limitations.

2.4 Toy Example of TreeSSS

Let N, t be positive integers and $\mathsf{SS}(N,t)$ a t-out-of-N Shamir's secret sharing scheme. Let $\mathsf{sk} \in \mathbb{Z}_q$ be a secret that we want to share. The purpose of this section is to concretely provide how to construct $\mathsf{SS}(5,3)$ using $\mathsf{SS}(3,2)$. That is, we will build a secret sharing scheme that sk should be only recovered when three or more parties gather.

Given a secret sk, we splits sk into $\mathsf{sk}_1^{(1)}, \mathsf{sk}_2^{(1)}, \mathsf{sk}_3^{(1)}$ by applying $\mathsf{SS}(3,2)$, where two secret shares can be used to recover secret sk. Then, applying $\mathsf{SS}(3,2)$ repeatedly, we divide $\mathsf{sk}_i^{(1)}$ into $\{\mathsf{sk}_{3i-2}^{(2)}, \mathsf{sk}_{3i-1}^{(2)}, \mathsf{sk}_{3i}^{(2)}\}$ and $\mathsf{sk}_j^{(2)}$ into $\{\mathsf{sk}_{3j-2}^{(3)}, \mathsf{sk}_{3j-1}^{(3)}, \mathsf{sk}_{3j}^{(3)}\}$. Now, we distribute secret shares $\{\mathsf{sk}_j^{(3)}\}_{j\in\{1,\cdots,27\}}$ to 5 parties as follows[11]:

$$\begin{aligned} &P_1 : \{\mathsf{sk}_1^{(3)}, \mathsf{sk}_6^{(3)}, \mathsf{sk}_{11}^{(3)}, \mathsf{sk}_{16}^{(3)}, \mathsf{sk}_{21}^{(3)}, \mathsf{sk}_{26}^{(3)}\}, P_2 : \{\mathsf{sk}_3^{(3)}, \mathsf{sk}_8^{(3)}, \mathsf{sk}_{13}^{(3)}, \mathsf{sk}_{18}^{(3)}, \mathsf{sk}_{23}^{(3)}\}, \\ &P_3 : \{\mathsf{sk}_2^{(3)}, \mathsf{sk}_7^{(3)}, \mathsf{sk}_{12}^{(3)}, \mathsf{sk}_{17}^{(3)}, \mathsf{sk}_{22}^{(3)}, \mathsf{sk}_{27}^{(3)}\}, P_4 : \{\mathsf{sk}_4^{(3)}, \mathsf{sk}_9^{(3)}, \mathsf{sk}_{14}^{(3)}, \mathsf{sk}_{19}^{(3)}, \mathsf{sk}_{24}^{(3)}\}, \\ &\qquad\qquad\qquad\qquad\qquad\qquad P_5 : \{\mathsf{sk}_5^{(3)}, \mathsf{sk}_{10}^{(3)}, \mathsf{sk}_{15}^{(3)}, \mathsf{sk}_{20}^{(3)}, \mathsf{sk}_{25}^{(3)}\}. \end{aligned}$$

In this case, we observe that any three parties can recover the secret and any two parties can not recover the secret sk.

For example, $\{P_2, P_4, P_5\}$ can reconstruct $\{\mathsf{sk}_2^{(2)}, \mathsf{sk}_3^{(2)}, \mathsf{sk}_5^{(2)}, \mathsf{sk}_7^{(2)}, \mathsf{sk}_8^{(2)}\}$ using their own secret shares. Consequently, they can also derive $\{\mathsf{sk}_1^{(1)}, \mathsf{sk}_3^{(1)}\}$. Finally, using $\{\mathsf{sk}_1^{(1)}, \mathsf{sk}_3^{(1)}\}$, they are able to recover the secret sk.

In contrast, $\{P_1, P_3\}$ can only reconstruct $\{\mathsf{sk}_1^{(2)}, \mathsf{sk}_4^{(2)}, \mathsf{sk}_6^{(2)}, \mathsf{sk}_9^{(2)}\}$, limiting them to the recovery of $\{\mathsf{sk}_2^{(1)}\}$ only. Thus, they are unable to recover the secret sk. Moreover, due to the privacy property of Shamir's secret sharing, $\{P_1, P_3\}$ cannot get any information about the secret sk. This observation will be used later in the proof of the privacy properties of our TreeSSS.

2.5 TreeSSS Meets Fully Homomorphic Encryption

We further remark that the simple replacement of a secret sharing scheme has an unexpected impact on the simulation security proof of TFHE. We recall the

[11] In this case, we define a specific partition that may not appear to be randomly distributed. However, if we sufficiently repeat the process of secret key distribution, Lemma 2.3 assures us that a linear secret sharing scheme for a threshold structure can be successfully constructed, provided that the secret shares are distributed randomly among parties.

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[12]:

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

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 TreeSSS approach, the coefficients are product of 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 [38], 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 [1,49]. The result is a new lemma which completes the security proof for the proposed construction. The proof of the lemma can be found in Suppl. B.2.

Lemma 2.4 ([14]) *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 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.

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

3 Preliminaries

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.

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:

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

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

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

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

$D_1 \approx_s D_2$ means that the distribution D_1 is statistically indistinguishable from distribution D_2.

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 [4, 7, 45]) *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 [17, 20, 22].

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 uniformly at random over $\mathbb{Z}_q^{m \times n}$, $\mathbf{s}$ is chosen from $\mathbb{Z}_q^n$, $\mathbf{e}$ is chosen from χ^m, and $\mathbf{u}$ is randomly chosen from $\mathbb{Z}_q^m$.

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

Definition 3.2 (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* $x = x_1 x_2 \dots x_N$

4 The Tree Secret Sharing Scheme for t-out-of-N threhold function

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

4.1 Preliminaries for Secret Sharing

This section provides several relevant definitions for secret sharing schemes as a representative. For this purpose, we adopt definitions/notations from [14].

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

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

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

- $\mathsf{SS.Share}(\mathsf{sk}, \mathbb{A}_{N,t})$*: There exists a share matrix* $\mathbf{M} \in \mathbb{Z}_q^{d \times n}$ *with positive integers* d, n *and associate a partition* T_i *of* $[d]$ *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* $S \in \mathbb{A}_{N,t}$*, 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.

A linear secret sharing scheme must satisfy the following properties.

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

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

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

4.2 TreeSSS from Shamir's Secret Sharing

We propose the Tree Secret Sharing Scheme for t-out-of-N threshold structure (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 [34,36].

This new method will allow us to construct t-out-of-N threshold access structure from Shamir's secret sharing for s-out-of-ℓ threshold functions, where $s \ll N$ and $\ell = 2s - 1$.

We first provide how to build TreeSSS for $\frac{N+1}{2}$-out-of-N threshold functions and extend it to arbitrary t-out-of-N threshold access structure. As in the standard syntax of secret sharing scheme Definition 4.2, TreeSSS consists of two algorithms, called TreeSSS.Share and TreeSSS.Combine.

Prior to introducing the TreeSSS, we first examine the SS.Share as outlined in Definition 4.2. The final step of this algorithm involves generating a vector

$$(\mathsf{share}_1, \ldots, \mathsf{share}_\ell)^T = \mathbf{M} \cdot (\mathsf{sk}, r_2, \ldots, r_n)^T,$$

where $\mathbf{M} \in \mathbb{Z}_q^{d\times n}$. Each party P_i receives a secret share comprised of the set $\{\mathsf{share}_j\}_{j\in T_i}$, where T_i is partition of the index set $[d]$ corresponding to party P_i.

It should be noted that a party P_i has the set $\{\mathsf{share}_j\}_{j\in T_i}$ can be regarded such that d secret shares are uniformly distributed to each party via a specific distribution scheme, without any overlapping of the shares share_j. In light of this, throughout the remainder of this paper, we will commonly employ the phrase 'uniformly distribute secret shares' or 'distribute secret shares'.

We now turn our attention to TreeSSS. The algorithm TreeSSS.Share consists of two primary steps:

1. For a predefined parameter L and the secret sk that wants to share, run the TreeSS algorithm (Algorithm 1) to generate level-L secret shares, and

2. Distribute all level-L secret shares among the parties. Here, *distribute* refers to conducting $\ell^L (= \mathsf{poly}(N))$ experiments as follows: For i-th experiment with $i \in [\ell^L]$, a distributor randomly (allowing repetition) samples $k \leftarrow [N]$ and add a level-L secret share $\mathsf{share}_i^{(L)}$ to a set sk_{P_k}. After ℓ^L experiments, a distributor forwards the set of shares $\mathsf{sk}_{P_k} = \{\mathsf{share}_i^{(L)}\}_i$ to each corresponding party P_k.

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 the unique level-0 secret share, denoted by $\mathsf{share}_1^{(0)}$. For each $0 \leq i \leq L$, one can generates level-i secret shares from level-$(i-1)$ secret shares as follows: The level-$(i-1)$ secret shares, $\mathsf{share}_j^{(i-1)}$, are split into level-i secret shares, $\{\mathsf{share}_k^{(i)}\}_{k \in \{\ell \cdot (j-1)+1, \cdots, \ell \cdot j\}}$. This process is repeated until level-L secret shares, $\{\mathsf{share}_j^{(L)}\}_{j \in [\ell^L]}$, are obtained and distributed randomly to each party. The details of the TreeSS algorithm is provided in Algorithm 1.

The other algorithm, $\mathsf{TreeSSS.Combine}$, is to repeatedly reconstruct level-i secret shares from level-$(i+1)$ secret shares. This process is repeated until we obtain the level-0 share sk. The correctness of $\mathsf{TreeSSS.Combine}$ is described in Theorem 4.6.

Algorithm 1: L-TreeSS Algorithm

```
Input  : Parties $P = \{P_1, \dots, P_N\}$ associated with a partition $T_i \subset [\ell^L]$ with
         $\ell = 2s - 1$ and $s \geq 3$,
         Shamir's secret sharing scheme for $s$-out-of-$\ell$ threshold access
structure $\mathsf{SS}(\ell, s)$,
         Secret key $\mathsf{sk}$,
Output: Output the set of secret shares $\{\mathsf{share}_j\}_{j \in [\ell^L]}$.
($\mathsf{share}_1^{(1)}, \cdots, \mathsf{share}_\ell^{(1)}) \leftarrow \mathsf{SS}(\ell, s)(\mathsf{sk})$
for $i = 2, \cdots, L$ do
    for $j = 1, \cdots, \ell^{i-1}$ do
        $(\mathsf{share}_{\ell \cdot (j-1)+1}^{(i)}, \cdots, \mathsf{share}_{\ell \cdot (j-1)+\ell}^{(i)}) \leftarrow \mathsf{SS}(\ell, s)(\mathsf{share}_j^{(i-1)})$
    end for
end for
return A set of secret shares $\{\mathsf{share}_j\}_{j \in [\ell^L]}$.
```

Construction 4.5 (L-TreeSSS) Let N be an odd integer and s be a small positive integer such that $s \ll N$. Let $\ell = 2s - 1$. Let $P = \{P_1, \cdots, P_N\}$ be a set of parties and $\mathsf{sk} \in \mathbb{Z}_q$ be a secret key. Given Shamir's secret sharing scheme $\mathsf{SS}(\ell, s)$ and iteration number L, a tree secret sharing scheme for iteration number L is a tuple of PPT algorithms $L\text{-}\mathsf{TreeSSS} = (\mathsf{TreeSSS.Share}, \mathsf{TreeSSS.Combine})$ that satisfies the following properties:

- The share algorithm $\mathsf{TreeSSS.Share}(\mathsf{sk}, \mathsf{SS}(\ell, s), L)$ takes as input a secret key sk, an Shamir's secret sharing $\mathsf{SS}(\ell, s)$ and an iteration number L, and construct a new secret sharing scheme $\mathsf{TreeSSS}_L$ 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 $B = \{\mathsf{share}_j^{(L)}\}_{j \in \bigcup_{P_i \in S} T_i}$, where $T_i = \{j \mid P_i \text{ has a secret share } \mathsf{share}_j^{(L)}\}$. 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:

$$\begin{aligned}
\hat{\mathsf{sk}} &= \sum_{j_L \in T^{(L)}} \lambda_{j_1}^{S_{j_1}^{(1)}} \cdots \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \lambda_{j_L}^{S_{j_L}^{(L)}} \cdot \mathsf{share}_{j_L}^{(L)} \\
&= \sum_{j_{L-1} \in T^{(L-1)}} \lambda_{j_1}^{S_{j_1}^{(1)}} \cdots \lambda_{j_{L-2}}^{S_{j_{L-2}}^{(L-2)}} \cdot \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \mathsf{share}_{j_{L-1}}^{(L-1)} \\
&= \cdots \\
&= \sum_{j_2 \in T^{(2)}} \lambda_{j_1}^{S_{j_1}^{(1)}} \cdot \lambda_{j_2}^{S_{j_2}^{(2)}} \cdot \mathsf{share}_{j_2}^{(2)} \\
&= \sum_{j_1 \in T^{(1)}} \lambda_{j_1}^{S_{j_1}^{(1)}} \cdot \mathsf{share}_{j_1}^{(1)} \\
&= \mathsf{share}_1^{(0)} = \mathsf{sk}
\end{aligned}$$

  where j_k is an index of level-k secret shares which uses $\mathsf{share}_{j_L}^{(L)}$ 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 $\mathsf{SS}(\ell, s)$. If a set of share B can not recover a level-0 secret share, then the combine algorithm outputs $\perp$.

Through Lemma 2.2, we prove that L-TreeSSS is a linear secret sharing scheme for $\frac{N+1}{2}$-out-of-N threshold structure, where L is sufficiently large.

Theorem 4.6 *Let N be an odd number and $P = \{P_1, \cdots, P_N\}$ be a set of parties. Let s be a small positive integer such that $s \ll N$, and $\ell = 2s - 1$. Given Shamir's secret sharing $\mathsf{SS}(\ell, s)$ and the iteration number $L \geq \log_{c_s} N + \log_s N + O(1)$ where $c_s = \frac{\ell}{2^{\ell-1}} \cdot \binom{\ell-1}{s-1}$, L-TreeSSS satisfies the correctness and privacy with $\frac{N+1}{2}$-out-of-N threshold structure with at least 1/2 success probability and the number of secret shares is $\ell^L = O(N^{\log_{c_s} \ell + \log_s \ell})$.*

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

Correctness. The correctness automatically holds underlying the following fact:

- The large-input majority circuit $\mathsf{MAJ}_N : \{0,1\}^N \to \{0,1\}$ can be built from the small-input majority circuit MAJ_ℓ when the iteration number $L \geq \log_{c_s} N + \log_s N + O(1)$ where $c_s = \frac{\ell}{2^{\ell-1}} \cdot \binom{\ell-1}{s-1}$ (Lemma 2.2).
- MAJ_ℓ corresponds to Shamir's secret sharing $\mathsf{SS}(\ell, s)$ for s-out-of-ℓ threshold access structure.

On top of the above facts, the total number of secret shares in the TreeSSS would be $\ell^L = O(N^{\log_{c_s} \ell + \log_s \ell}) \approx O(N^{3+2.3/\log s})$.[13]

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

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

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

To this end, $S_b^{(k+1)}$ can be divided into two subsets:

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

Note that by definition, level-$(k+1)$ secret shares $S_{b,P}^{(k+1)}$ come from level-k secret shares in $S_b^{(k)}$ by using Shamir's secret sharing for every b. Since nobody

[13] The detailed computation of approximations will be given by Suppl. E.

can distinguish between $S_0^{(k)}, S_1^{(k)}$ by the assumption, $S_{0,P}^{(k+1)}, S_{1,P}^{(k+1)}$ also follow the identical distribution.

Moreover, $S_{0,I}^{(k+1)}, S_{1,I}^{(k+1)}$ comes from Shamir's secret sharing and cannot recover level-k 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 $S_b^{(k+1)} = S_{b,P}^{(k+1)} \bigcup S_{b,I}^{(k+1)}$.

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

□

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

Construction 4.5 indicates that there is TreeSSS for $\frac{N+1}{2}$-out-of-N threshold access structure for an odd N. In this section, we generate TreeSSS for t-out-of-N threshold access structure, for any integers t and N.

For simple description, we denote TreeSSS for t-out-of-N threshold access structure by $\mathsf{TreeSSS}(N,t)$. Assume that there exists $\mathsf{TreeSSS}(N, \frac{N+1}{2})$ for an odd integer N.

Case 1) $\mathsf{TreeSSS}(N,t)$ **for** $t > \frac{N+1}{2}$ **and odd** N**.** According to Construction 4.5, one first generates $\mathsf{TreeSSS}(2t-1,t)$. Since $2t-1 > N$, we disregard the secret shares beyond those distributed to the N parties.

Case 2) $\mathsf{TreeSSS}(N,t)$ **for** $t < \frac{N+1}{2}$ **and odd** N**.** Let $r = N - 2t + 1$. Then, $N+r$ is always odd and satisfies $\frac{N+r+1}{2} = t+r$. According to Construction 4.5, one can make $\mathsf{TreeSSS}(N+r, t+r)$ where r satisfies $\frac{N+r+1}{2} = t+r$. 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) $\mathsf{TreeSSS}(N,t)$ **for any** t **and** N **is even.** According to Construction 4.5, one can generate $\mathsf{TreeSSS}(N+1,t+1)$, which automatically achieves the goal.

Remark that for each case, the total number of parties is less than $2N$. Therefore, the number of secret shares is still $O(N^{\log_{c_s}\ell + \log_s \ell})$ with constant integer s and $\ell = 2s-1$. As a result, we can generate $\mathsf{TreeSSS}(N,t)$ for arbitrary t and N while preserving the number of secret shares.

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 [14].

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 $\mathbb{A}_{N,t}$ be a threshold structures on P. A threshold fully homomorphic encryption scheme for $\mathbb{A}_{N,t}$ 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* $\mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_{N,t})$ *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* $\hat{\mu} \in \{0, 1, \perp\}$*.*

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* $\mathbb{A}_{N,t}$ *where* $1 \leq t \leq N$*.*
2. *The challenger* $\mathcal{C}$ *runs* $\mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_{N,t}) \rightarrow (\mathsf{pk}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N)$*, and gives* pk *to* $\mathcal{A}$*.*

3. *$\mathcal{A}$ outputs a set $S \subset \{P_1, \cdots, P_N\}$ such that $S \notin \mathbb{A}_{N,t}$.*
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 $\mathbb{A}_{N,t}$, 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 $\mathbb{A}_{N,t}$ where $1 \le t \le N$.*
2. *The challenger $\mathcal{C}$ runs $\mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_{N,t}) \to (\mathsf{pk}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N)$, 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 $\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A}_{N,t}) \to (\mathsf{pk}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N, \mathsf{st})$, 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 A.11) with noise bound B and multiplicative constant $((2s-1)!)^L$ where $L \ge \log_{c_s} N + \log_s N + O(1)$ and c_s is $\frac{2s-1}{4^{s-1}} \cdot \binom{2s-2}{s-1}$ for a positive integer $s \ge 2$.

- Let $\mathsf{TreeSSS}$ be a level-L tree secret sharing scheme built from s-out-of-$2s-1$ Shamir's secret sharing scheme (Section 4.2).

The construction presented in this paper is similar to the one in [14], except that we utilize a TreeSSS as opposed to a $\{0,1\}$-LSSS instantiated by [38]. 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:

- $(\mathsf{pk}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_{N,t})$:
  1. Sample $(\mathsf{fhepk}, \mathsf{fhesk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$.
  2. Compute $(\mathsf{share}_1^{(L)}, \cdots, \mathsf{share}_{(2s-1)^L}^{(L)}) \leftarrow \mathsf{TreeSSS.Share}(\mathsf{fhesk}, \mathbb{A}_{N,t})$.
  3. Distribute the secret shares to each party P_i and define a index set of each party $T_i := \{j \mid P_i \textit{ has } \mathsf{share}_j^{(L)}\}$.
  4. Return $\mathsf{pk} = \mathsf{fhepk}$ and $\mathsf{sk}_i = \{\mathsf{share}_j^{(L)}\}_{j \in T_i}$ 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
  $$\hat{\mathbf{p}}_j^{(L)} = \mathsf{FHE.Dec}_0(\mathsf{share}_j^{(L)}, \mathsf{ct}) + ((2s-1)!)^L e_j \in \mathbb{Z}_q$$
  for every $j \in T_i$.
  2. Return $p_i = \{\hat{\mathbf{p}}_j^{(L)}\}_{j \in T_i}$ as its partial decryption.
- $\hat{\mu} \leftarrow \mathsf{TFHE.FinDec}(\mathsf{pk}, B)$:
  1. Check if $S \in \mathbb{A}_{N,t}$ or not: If $S \notin \mathbb{A}_{N,t}$, return $\perp$.
  2. If $S \in \mathbb{A}_{N,t}$, compute a minimal valid share set $T \subset \cup_{i \in S} T_i$ and $\mu \leftarrow \mathsf{FHE.Dec}_1(\sum_{j \in T} c_j^S \cdot \hat{\mathbf{p}}_j^{(L)})$.
  3. Return $\hat{\mu}$.

Theorem 5.7 ([14]) *Suppose FHE is a compact fully homomorphic encryption scheme. Then, the TFHE scheme in 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* $\mathsf{TreeSSS}$ *is a level-L tree secret sharing scheme, where $L \geq \log_{c_s} N + \log_s N + O(1)$, in Section 4.2 that satisfies the correctness. Then, TFHE scheme in Construction 5.6 with respect to parameter regime B_{sm} such that $B + ((2s-1)!)^{2L} \cdot (2s-1)^L \cdot B_{sm} \leq \lfloor \frac{q}{4} \rceil$ satisfies the correctness.*

Theorem 5.9 (Adopt from [14]) *Suppose FHE is a fully homomorphic encryption scheme that satisfies security and* $\mathsf{TreeSSS}$ *is a tree secret sharing scheme that satisfies the correctness. Then, TFHE scheme from Construction 5.6 satisfies semantic security.*

Theorem 5.10 *Suppose* FHE *is a fully homomorphic encryption scheme that satisfies security and* TreeSSS *is a tree secret sharing scheme that satisfies correctness and privacy. Then,* 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 [14] 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 $\{\mathsf{share}_j^{(L)}\}_{j \in T^*}$ 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 $\{\mathsf{share}_j^{(L)}\}_{j \in T^*}$ 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 $p_i = \{\hat{\mathbf{p}}_j^{(L)}\}_{j \in T_i}$, based on the following conditions:

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

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

  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 $\sum_{k \in \bar{T}} c_k \cdot \mathsf{share}_k^{(L)} = \mathsf{fhesk}$. Then, $\mathcal{C}$ returns

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

  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 $\hat{\mathbf{p}}_j^{(L)}$ 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 $\hat{\mathbf{p}}_j^{(L)}$

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

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 2.4 that $|(c_j)^{-1} \cdot \tilde{e}| \leq ((2s-1)!)^{2L} \cdot B$. 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 say 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 [14] through the following theorems. Throughout this section, we adopt the definitions and theorems from [14] that provides the concept of UT and the first construction.

6.1 Definitions

This section presents definitions and properties of the universal thresholdizer. We basically follow presentations of the original paper [14].

Definition 6.1 (Universal Thresholdizer [14]) *Let $P = \{P_1, \cdots, P_N\}$ be a set of parties and $\mathbb{A}_{N,t}$ be a threshold structures on P. A universal thresholdizer scheme for $\mathbb{A}_{N,t}$ 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, $\mathsf{UT.Setup}(1^\lambda, d, \mathbb{A}_{N,t}, \mathbf{x})$ takes as input the security parameter λ, a depth bound d, a threshold structure $\mathbb{A}_{N,t}$ 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.*

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

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 $(\mathsf{pp}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}_{N,t}, \mathbf{x})$ 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 $\mathbb{A}_{N,t}$*
2. *The challenger $\mathcal{C}$ runs $(\mathsf{pp}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}_{N,t}, \mathbf{x})$ 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 $\mathbb{A}_{N,t}$, 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 $\mathbb{A}_{N,t}$ and a message $\mathbf{x} \in \{0,1\}^k$.*
2. *The challenger runs $\mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}_{N,t}, \mathbf{x}) \rightarrow (\mathsf{pp}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N)$, 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 $\mathbb{A}_{N,t}$ and a message $\mathbf{x} \in \{0,1\}^k$.*
2. *The challenger runs $\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A}_{N,t}) \rightarrow (\mathsf{pp}, \mathsf{sk}_1, \cdots, \mathsf{sk}_N, \mathsf{st})$, 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 A.12) [39, 47] and non-interactive commitment (Definition A.14) scheme [11]. We again note that the description is based on the original paper [14].

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

- $\mathsf{UT.Setup}(1^\lambda, d, \mathbb{A}_{N,t}, \mathbf{x})$:
  1. Sample TFHE keys $(\mathsf{tfhepk}, \{\mathsf{tfhesk}\}_{i \in [N]}) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_{N,t})$.
  2. Sample $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Enc}(\mathsf{tfhepk}, x_i)$ for $i = 1, \ldots, k$.
  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 $i = 1, \ldots, N$.

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 $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{tfhepk}, C, \mathsf{ct}_1, \mathsf{ct}_2, \ldots, \mathsf{ct}_k)$
  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 [14].

Theorem 6.8 ([14]) *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. Benny Applebaum, Oded Nir, and Benny Pinkas. How to recover a secret with o(n) additions. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 236–262, Cham, 2023. Springer Nature Switzerland.
4. 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.
5. 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.
6. Marshall Ball, Alper Çakan, and Tal Malkin. Linear Threshold Secret-Sharing with Binary Reconstruction. In Stefano Tessaro, editor, *2nd Conference on Information-Theoretic Cryptography (ITC 2021)*, volume 199 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:22, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
7. Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In *Theory of Cryptography Conference*, pages 201–218. Springer, 2010.
8. Fabrice Benhamouda, Shai Halevi, and Lev Stambler. Weighted Secret Sharing from Wiretap Channels. In Kai-Min Chung, editor, *4th Conference on Information-Theoretic Cryptography (ITC 2023)*, volume 267 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:19, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
9. Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. *Journal of Cryptology*, 33(2):459–493, 2020.
10. G. R. Blakley and Catherine Meadows. Security of ramp schemes. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, pages 242–268, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.
11. Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 15(1):23–27, 1983.
12. 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.
13. 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.
14. 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.
15. Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from lwe with polynomial modulus. *Cryptology ePrint Archive*, 2023.
16. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Advances in Cryptology – CRYPTO 2012*, pages 868–886, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

17. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.
18. Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key fhe with short ciphertexts. In *Annual Cryptology Conference*, pages 190–213. Springer, 2016.
19. 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.
20. 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.
21. 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.
22. 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.
23. 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.
24. 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.
25. 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.
26. Yvo Desmedt and Yair Frankel. Threshold cryptosystesns. *Advance in Cryptology*, pages 305–315, 1989.
27. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144, 2012.
28. 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.
29. Sanjam Garg, Abhishek Jain, Pratyay Mukherjee, Rohit Sinha, Mingyuan Wang, and Yinuo Zhang. Cryptography with weights: Mpc, encryption and signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 295–327, Cham, 2023. Springer Nature Switzerland.
30. 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.
31. Rosario Gennaro and Steven Goldfeder. One round threshold ecdsa with identifiable abort. *Cryptology ePrint Archive*, 2020.
32. Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security. In *Interna-*

tional Conference on Applied Cryptography and Network Security, pages 156–174. Springer, 2016.

33. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold dss signatures. *Information and Computation*, 164(1):54–84, 2001.
34. Oded Goldreich. *On (Valiant's) Polynomial-Size Monotone Formula for Majority*, pages 17–23. Springer International Publishing, Cham, 2020.
35. 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.
36. Arvind Gupta and Sanjeev Mahajan. Using amplification to compute majority with small majority gates. *Computational Complexity*, 6(1):46–63, 1996.
37. Shlomo Hoory, Avner Magen, and Toniann Pitassi. Monotone circuits for the majority function. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 410–425. Springer, 2006.
38. Aayush Jain, Peter MR Rasmussen, and Amit Sahai. Threshold fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2017:257, 2017.
39. 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.
40. 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.
41. 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.
42. Yehuda Lindell. Fast secure two-party ecdsa signing. In *Annual International Cryptology Conference*, pages 613–644. Springer, 2017.
43. 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.
44. Philip MacKenzie and Michael K Reiter. Two-party generation of dsa signatures. *International Journal of Information Security*, 2(3):218–239, 2004.
45. 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.
46. Chris Peikert and Sina Shiehian. Multi-key fhe from lwe, revisited. In *Theory of Cryptography Conference*, pages 217–238. Springer, 2016.
47. 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.
48. Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
49. Victor Shoup. Practical threshold signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 207–220. Springer, 2000.
50. 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.

51. Leslie G. Valiant. Short monotone formulae for the majority function. *Journal of Algorithms*, 5(3):363–366, 1984.
52. 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

A Missing Preliminaries

A.1 Majority Circuits

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

Definition A.1 (Monotone Boolean formula [14]) *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 A.2 (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 [34, 36, 51]. However, please refer to the original papers for a full understanding and proof of these results.

Using Lemma 2.3, Valiant proposed a (recursive) construction of monotone Boolean formula for computing the majority function. The unit circuit $\varphi^{(i+1)} = (\varphi_1^{(i)} \wedge \varphi_2^{(i)}) \vee (\varphi_3^{(i)} \wedge \varphi_4^{(i)})$ is repeated to construct the majority function for N. [51] showed that repeating the unit circuit up to level-$2.65 \log N$ can construct the majority function for N.

Lemma A.3 ([51]) *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 one can create an $O(N^{5.3})$-size monotone formula for computing the majority function MAJ_N with at least 1/2 success probability.*

Note that the construction presented in Lemma A.4 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 [51]. 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 A.4 ([34]) *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 $O(N^7)$.*

A.2 Shamir's secret sharing scheme

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

Definition A.5 (Shamir's secret sharing [48]) *Let $P = \{P_1, \cdots, P_N\}$ be a set of parties and let $\mathbb{A}_{N,t}$ 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:*

- *$\mathsf{SS.Share}(\mathsf{sk}, \mathbb{A}_{N,t}) \to (\mathsf{sk}_1, \cdots, \mathsf{sk}_N)$: 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 $\mathbb{A}_{N,t}$ 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:*

$$\sum_{P_i \in S} \lambda_i^S \cdot \mathsf{sk}_i = (1, 0, \cdots, 0), \text{ where } \lambda_i^S := \prod_{P_j \in S \setminus \{P_i\}} \frac{-j}{i - j}.$$

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

Theorem A.6 ([48]) *Let $P = \{P_1, \cdots, P_N\}$ be a set of parties and let $\mathbb{A}_{N,t}$ 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*

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

A.3 Fully Homomorphic Encryption

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

Definition A.7 (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 A.7.

Definition A.8 (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 A.9 (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 A.10 (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 A.11 (Special FHE [14]) *We say that a* FHE *scheme is a special* FHE *scheme 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*. This* c *is called the multiplicative constant.*
  - $\mu \leftarrow \mathsf{Dec}_1(p)$*: 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*.*

A.4 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 [14].

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

Definition A.12 (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 A.13 ([39, 47]) *PZK can be constructed from one-way functions.*

We now introduce a new component for the universal thresholdizer, as described in [9, 14].

Definition A.14 (Non-interactive Commitment [11]) *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 A.15 ([11]) *A non-interactive commitment can be built from injective one-way functions.*

B Proofs in Section 2

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

Lemma B.1 ([36]) *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. $p' := \Pr[\mathsf{MAJ}_s(X_1, \cdots, X_{2s-1}) = 1] = \sum_{j=0}^{s-1} \binom{2s-1}{j} \cdot p^{2s-1-j}(1-p)^j$.
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. $1 - p' < \binom{2s-1}{s}(1-p)^s$.

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.

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

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

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 [21], 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. □

B.2 Proof of Lemma 2.4

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 $(N!)^2$.

C Proofs in Section 4

C.1 Proof of Section 2.3

In Section 2.3, we argue that if $L \geq \log_{c_s} N + \log_s (N+\kappa) + O(1)$, the success probability of constructing a share matrix is larger than or equal to $1 - 2^{-\kappa}$.

The failure probability of constructing a share matrix is at most

$$\sum_{x \in \{0,1\}^N} Pr[F(x) \neq \mathsf{MAJ}(x)],$$

where the function F outputs 0 if the secret can not recover and 1 if the secret can recover when parties of indexes 1 in input x are gathered. Similar to Lemma 2.3, if each probability $Pr[F(x) \neq \mathsf{MAJ}(x)]$ is less than $2^{-N-\kappa-1}$, then the success probability of constructing a share matrix is larger than $1 - 2^{-\kappa}$.

According to Theorem B.1, to make each probability $Pr[F(x) \neq MAJ(x)]$ be less than 2^{-N-1}, the iteration number is $\log_{c_s} N + O(1) + \log_s N$. If we replace the last part of iteration number $\log_s N$ with $\log_s(N+\kappa)$, each probability becomes less than $2^{-N-\kappa-1}$. □

D Proofs in Section 5

The proofs are almost the same as the original proofs in the full version of [14].

D.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. □

D.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:

$$(\mathsf{share}_1^{(L)}, \cdots, \mathsf{share}_{(2s-1)^L}^{(L)}) \leftarrow \mathsf{TreeSSS.Share}(\mathsf{fhesk}, \mathbb{A}_t).$$

- The setup algorithm returns $\mathsf{pk} = \mathsf{fhepk}$ and $\mathsf{sk}_i = \{\mathsf{share}_j^{(L)}\}_{j \in T_i}$ for $i \in [N]$.
- The partial decryption algorithm outputs $p_i = \{\hat{\mathbf{p}}_j{}^{(L)}\}_{j \in T_i}$, where

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

for every $j \in T_i = \{j \mid P_i \text{ } has \text{ } \mathsf{share}_j^{(L)}\}$ and any (valid) ciphertext ct.

Let $T^{(L)} \subseteq \cup_{i \in S} T_i$ be the minimal valid share set for $S \in \mathbb{A}_t$. Then, $\{\mathsf{share}_j^{(L)}\}_{j \in T^{(L)}}$ can be recovered to the secret key sk using $\mathsf{TreeSSS}$ built from $\mathsf{SS}(2s-1, s)$ proposed by Section 4.2.

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

$$\begin{aligned}\mathsf{fhesk} &= \mathsf{share}^{(0)} \\ &= \sum_{j_1 \in T^{(1)}} \lambda_{j_1}^{S_{j_1}^{(1)}} \cdot \mathsf{share}_{j_1}^{(1)}\end{aligned}$$

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

where j_k is an index of level-k secret shares which uses $\mathsf{share}_j^{(L)}$ 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 $\mathsf{SS}(2s-1, s)$.

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

$$
\begin{aligned}
&\sum_{j_L \in T^{(L)}} \lambda_{j_1}^{S_{j_1}^{(1)}} \cdots \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \lambda_{j_L}^{S_{j_L}^{(L)}} \cdot \hat{\mathbf{p}}_{j_L}^{(L)} \\
&= \sum_{j_L \in T^{(L)}} \lambda_{j_1}^{S_{j_1}^{(1)}} \cdots \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \lambda_{j_L}^{S_{j_L}^{(L)}} \cdot \left(\mathsf{FHE.Dec}_0(\mathsf{share}_{j_L}^{(L)}, \mathsf{ct}) + ((2s-1)!)^L e_{j_L}\right) \\
&= \mathsf{FHE.Dec}_0\left(\sum_{j_L \in T^{(L)}} \lambda_{j_1}^{S_{j_1}^{(1)}} \cdots \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \lambda_{j_L}^{S_{j_L}^{(L)}} \cdot \mathsf{share}_{j_L}^{(L)}, \mathsf{ct}\right) \\
&+ \sum_{j_L \in T^{(L)}} \lambda_{j_1}^{S_{j_1}^{(1)}} \cdots \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \lambda_{j_L}^{S_{j_L}^{(L)}} \cdot ((2s-1)!)^L e_{j_L} \\
&= \mathsf{FHE.Dec}_0(\mathsf{fhesk}, \mathsf{ct}) + \sum_{j_L \in T^{(L)}} \lambda_{j_1}^{S_{j_1}^{(1)}} \cdots \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \lambda_{j_L}^{S_{j_L}^{(L)}} \cdot ((2s-1)!)^L e_{j_L} \\
&= \mu \lfloor \frac{q}{2} \rceil + e + \sum_{j_L \in T^{(L)}} \lambda_{j_1}^{S_{j_1}^{(1)}} \cdots \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \lambda_{j_L}^{S_{j_L}^{(L)}} \cdot ((2s-1)!)^L e_{j_L}.
\end{aligned}
$$

Consequently, $\mathsf{FHE.Dec}_1(\sum_{j_L \in T^{(L)}} \lambda_{j_1}^{S_{j_1}^{(1)}} \cdots \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \lambda_{j_L}^{S_{j_L}^{(L)}} \cdot \hat{\mathbf{p}}_{j_L}^{(L)})$ returns the correct messages when the error term is appropriately bounded because of Definition A.11.

Let e_{sm} be a noise smudging error of the form

$$
\sum_{j_L \in T^{(L)}} \lambda_{j_1}^{S_{j_1}^{(1)}} \cdots \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \lambda_{j_L}^{S_{j_L}^{(L)}} \cdot ((2s-1)!)^L e_{j_L}.
$$

By Lemma 2.4, 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. □

D.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.

E About Approximation

In [21] and Suppl. B, we observe that c_s has the following property:

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

we can get following inequality:

$$\begin{aligned}
\log_{c_s}(2s-1) + \log_s(2s-1) &\le \log_{2\sqrt{\frac{s-1/2}{\pi}}}(2s-1) + \log_s(2s) \\
&\le (2 + \log_{2\sqrt{\frac{s-1/2}{\pi}}}(\pi/2)) + (1 + \log_s 2) \\
&\le 3 + 2\log_{\frac{4}{\pi}(s-1/2)}(\pi/2) + \frac{1}{\log s} \\
&\le 3 + \frac{1.30299}{\log s} + \frac{1}{\log s} \\
&\le 3 + \frac{2.30299}{\log s},
\end{aligned}$$

where $\frac{4}{\pi}(s-1/2) \ge s$ $(s \ge 2)$ and $\log_2 \frac{\pi}{2} = 0.65149612947$.

F Observation of $\{0,1\}$-LSSS with [51] construction

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 [14]. The use of monotone Boolean formulas [41] was proposed as an instantiation of $\{0,1\}$-LSSS. However, the polynomial-sized expression of threshold functions was proven by Valiant and Goldreich [34, 51]. Recently, [38] proposed using a folklore algorithm to demonstrate that monotone Boolean formulas are a part of $\{0,1\}$-LSSS. We briefly summarize the construction of threshold functions.

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 $i \ge 1$, 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 [34, 51], 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, [38] 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 [38] 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 [14, 38].

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. 4: Folklore Algorithm in [38].

F.1 Regarding $\{0,1\}$-LSSS as Tree Secret Sharing

We reinterpret a secret sharing algorithm for threshold functions by utilizing the iterative steps of Boolean formula construction described in [51]. This allows us to construct the share matrix $\mathbf{M}$ through iterative matrix multiplications.

[51] proves 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 5. 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

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

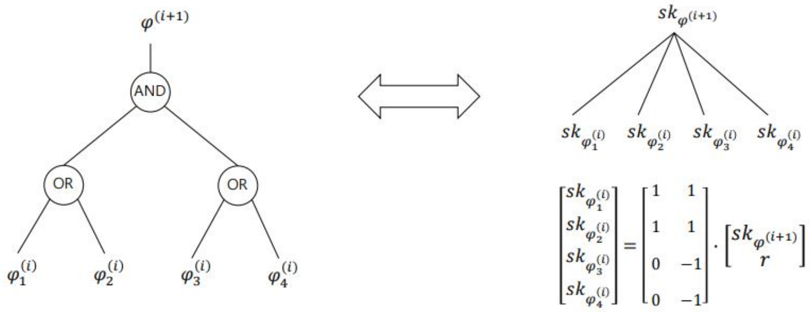

$\varphi^{(i+1)}$
AND
OR
OR
$\varphi_1^{(i)}$
$\varphi_2^{(i)}$
$\varphi_3^{(i)}$
$\varphi_4^{(i)}$
$sk_{\varphi^{(i+1)}}$
$sk_{\varphi_1^{(i)}}$
$sk_{\varphi_2^{(i)}}$
$sk_{\varphi_3^{(i)}}$
$sk_{\varphi_4^{(i)}}$
$\begin{bmatrix} sk_{\varphi_1^{(i)}} \\ sk_{\varphi_2^{(i)}} \\ sk_{\varphi_3^{(i)}}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

Fig. 5: 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 F.1 and we defer the proof in F.2.

Proposition F.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.

F.2 Proof of Proposition F.1

Let $\mathbf{S}^{(0)} = \mathsf{sk} \in \mathbb{Z}_q$ be a secret key that we want to share and $\mathbf{U}^{(0)} = \mathbf{D}$ be a share matrix. Let $\mathbf{R}^{(0)} = (\mathbf{S}^{(0)}, \mathbf{r}^{(0)})^T \in \mathbb{Z}_q^m$ be a vector where $\mathbf{r}^{(0)} = (r_2^{(0)}, \cdots, r_m^{(0)})$ is a random vector and its elements are uniformly sampled, $r_2^{(0)}, \cdots, r_m^{(0)} \leftarrow \mathbb{Z}_q$. By repeated matrix multiplications, we can generalize that for an integer $1 \le i \le L$, level-(i) secret shares $\mathbf{S}^{(i)}$, share matrix $\mathbf{U}^{(i)}$, and input vector $\mathbf{R}^{(i)}$ are as follows:

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

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

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

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

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

where $\mathbf{U}^{(0)} \cdot \mathbf{P}^{(0)} = \mathbf{D}$. Then, we show that $\mathsf{SS}^{(L)}$ is a linear secret sharing scheme of sk for all positive integer L. □