# 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 secret sharing, which lacks compactness and results in ciphertext sizes of $O(N \log N)$, where $N$ is the number of parties involved in the threshold system, 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 to $O(N^{3+o(1)})$ 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

The $t$-out-of-$N$ threshold cryptography [26, 27, 29] is a public key cryptosystem that enables the distribution of secret keys among $N$ parties. To obtain the

* Author list in alphabetical order; see https://www.ams.org/profession/leaders/culture/JointResearchandItsPublicationfinal.pdf.

plaintext from a ciphertext encrypted with this system, $t$ or more 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) [41, 49], 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: Shamir 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 $O(N \log N)$.[4]

The TFHE based on $\{0,1\}$-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 [35, 53], 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

We propose a communication-efficient UT by constructing an efficient linear secret sharing scheme for $t$-out-of-$N$ threshold access structure, called TreeSSS. The linear secret sharing scheme reduces the number of shared keys compared to $\{0,1\}$-LSSS in [14], leading to less secret shares being shared in the setup algorithm of TFHE. Thus, 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 not limited to threshold signatures [2], threshold multi-key FHE schemes [5], and decentralized Attribute-Based Encryption (ABE) [54].

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

TreeSSS is devised through the iterative application of $s$-out-of-$2s-1$ Shamir secret sharing scheme, where $s \ll N$. This result exactly matches the *optimal amplification* used to construct large input majority function from small input majority functions [37]. The results are given in Table 1.

Lastly, we address a previously overlooked probability issue that was not fully considered in the construction of the share matrix $\mathbf{M}$ using $\{0,1\}$-LSSS. This issue originates from Valiant's monotone Boolean formula in [53], which serves as the foundation structure for $\{0,1\}$-LSSS. According to Valiant's formula, as detailed in Lemma 3.4, there is a guarantee 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 properly constructed share matrix as intended.

To verify the correctness of $\mathbf{M}$, we need to check every case where the secret key is recoverable by $t$ or more parties, and not recoverable by less than $t$ parties. This verification requires $O(N^t) = O(\binom{N}{t})$ offline cost, leading to exponential time for large $t$. To address this, we adjust our construction to ensure the desired share matrix $\mathbf{M}$ is achieved with probability $1-2^{-\kappa}$, where $\kappa$ is a predetermined integer. Then, the base for the number of secret shares increases from $N$ to $N+\kappa$. Further details and modifications are discussed in Section 4.5.

| Secret Sharing Scheme | | Structure | $O(\log q)$ | # of keys |
|---|---|---|---|---|
| Previous | Shamir SS | $t$-out-of-$N$ | $O(N \log N)$ | $N$ |
| | $\{0,1\}$-LSSS[5] | | $O(\log N)$ | $O(N^{5.3})$ |
| TreeSSS | $\mathsf{SS}(3,2)$ | $t$-out-of-$N$ | $O(\log N)$ | $O(N^{4.3})$ |
| | $\mathsf{SS}(2s-1,s)$ | | $O(s \log s \cdot \log N)$ | $O(N^{3+\frac{2.3}{\log s}})$ |

Table 1: The comparison results between the previous TFHE and ours. The column 'structure' indicates the access structure of secret sharing schemes. $\mathsf{SS}(N,t)$ indicates Shamir 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.

*Remark 1 (Another Compiler).* [14, Section 8.4] additionally provides another compiler for constructing a compact UT based on non-compact UT $\mathsf{nUT}$ and a compact FHE $\mathsf{FHE}$. Here, non-compact UT is built from a non-compact TFHE $\mathsf{nTFHE}$ built from Shamir secret sharing. More precisely, they built a compact $\mathsf{TFHE}$ from $\mathsf{nUT}$ and $\mathsf{FHE}$, which implies the compact UT. However, we believe that this construction is quite infeasible in current $\mathsf{nTFHE}$ and $\mathsf{FHE}$ parameters.

[5] [14, 40] propose the definition of $\{0,1\}$-LSSS for arbitrary access structure, but they only instantiated $\{0,1\}$-LSSS for $t$-out-of-$N$.

To this end, we briefly introduce how to compile $\mathsf{TFHE}$ from $\mathsf{nUT}$ and $\mathsf{FHE}$. First of all, $\mathsf{TFHE.Setup}$ is instantiated by two parts: 1) sample $(\mathsf{fhepk}, \mathsf{fhesk}) \leftarrow \mathsf{FHE.Setup}$, and 2) sample $(\mathsf{utpp}, \{\mathsf{sk}_i\}_{i=1}^{N}) \leftarrow \mathsf{nUT.Setup}(\mathbb{A}_{N,t}, \mathsf{fhesk})$, where $\mathbb{A}_{N,t}$ is $t$-out-of-$N$ access structure. More precisely, $\mathsf{utpp} \subset \mathsf{TFHEpp}$ contains a set of ciphertexts $\{\mathsf{nct}_i\}_{i=1}^{k}$, where $k$ is the bit-length of $\mathsf{fhesk} \in \mathbb{Z}_q^n$. By definition, $\mathsf{fhesk}$ can be regarded as $n \log q$-bit string. In fact, for every $i$-th bit of $\mathsf{fhesk}_i$, $\mathsf{nct}_i \leftarrow \mathsf{nTFHE.Enc}(\mathsf{fhesk}_i)$ is a public parameter $\mathsf{TFHE.pp}$ of $\mathsf{TFHE}$. In summary, $\mathsf{TFHE.pp}$ at least consists of $n \log q$ $\mathsf{nTFHE}$ *ciphertexts*.

Fortunately, $\mathsf{TFHE.Enc}(\cdot)$ and $\mathsf{TFHE.Eval}(\cdot)$ are the identical to $\mathsf{FHE.Enc}(\cdot)$ and $\mathsf{FHE.Eval}(\cdot)$, respectively, even though $\mathsf{TFHE.Setup}$ takes a long time. However, by construction in [14, Section 8.4], $\mathsf{TFHE.ParDec}(\cdot)$ is infeasible. More precisely, $\mathsf{TFHE.ParDec}(\mathsf{sk}_i, \mathsf{ct})$ is to sample $\mathsf{p}_i \leftarrow \mathsf{nUT.Eval}(\mathsf{utpp}, \mathsf{sk}_i, C_{\mathsf{ct}})$, where a circuit $C_{\mathsf{ct}}$ satisfies $C_{\mathsf{ct}}(\mathsf{fhesk}) = \mathsf{FHE.Dec}(\mathsf{fhesk}, \mathsf{ct})$. Since $\mathsf{nUT.Eval}$ has to run $\mathsf{nTFHE.Eval}(C_{\mathsf{ct}}, \{\mathsf{nct}\}_{i=1}^{n \log q})$, this step is so far from the practicality, regardless of $(t, N)$.[6]

For example, according to the recent guidance of FHE parameters [15], to allow a depth 5 circuits for BGV FHE scheme, a parameter $(n, \log q) = (2^{14}, 424)$ is recommended to achieve 128-bit security. That is, to construct $\mathsf{TFHE}$ by combining $\mathsf{nUT}$ with BGV scheme, at least $2^{14} \cdot 424 \approx 2^{22}$ $\mathsf{nTFHE}$ ciphertexts are required to $\mathsf{TFHE.Setup}$. Last, one can instantiate a compact UT, $\mathsf{cUT}$ from $\mathsf{TFHE}$, PZK and a non-interactive commitment scheme.

Even more, this compiler performs more worse than TreeSSS-based TFHE and UT when $N$ is large since it needs $N$ times of $\mathsf{nTFHE.Eval}(C_{\mathsf{ct}}, \cdot)$, where it can be considered as a bootstrapping algorithm of $\mathsf{FHE}$. It is well known that bootstrapping algorithm is the most intensive part FHE schemes.

### 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 $\mathsf{TFHE}$ is entirely distinct from that of the recent papers [6, 16, 24, 42]. Our focus is on directly improving $\{0,1\}$-LSSS, while [16] uses it as is. We thus believe our TreeSSS and the technique in [16] can be combined to create an efficient TFHE and UT construction. We further note that [42] improved the bootstrapping technique of FHEW/TFHE to achieve threshold FHE, while [16, 24] both achieved a polynomial modulus-to-noise ratio TFHE from LWE using Rènyi divergence and the noise flooding technique. [24] was specialized for Torus-FHE [23], while [16] 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

[6] $\mathsf{nTFHE.Eval}(C_{\mathsf{ct}}, \{\mathsf{nct}_i\})$ can be regarded as bootstrapping step of $\mathsf{nTFHE}$.

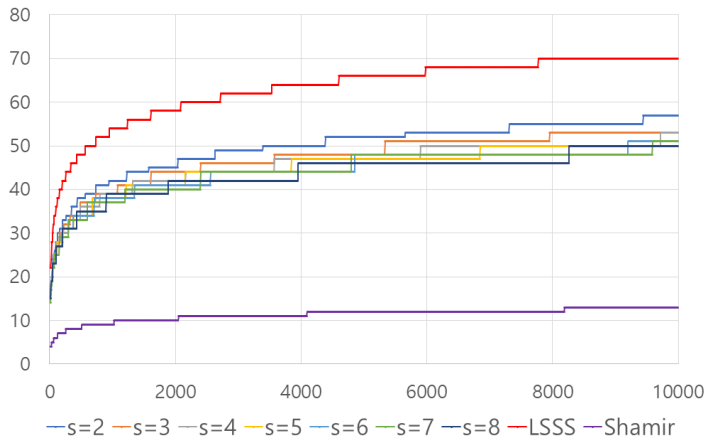
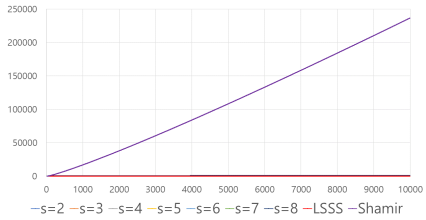
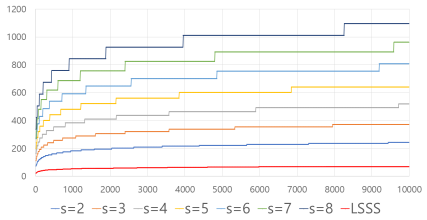



Fig. 1: Comparison the number of secret shares: Ours, $\{0,1\}$-LSSS, and Shamir secret sharing. Log-size of the number of secret shares according to number of parties $N$.




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




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

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

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 [35,37]. However, these constructions are not useful for threshold circuits similar to TFHE. For example, [35] 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 [53]. 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-$N$. [39] improved upon the result in [53], which had a size of $O(n^{1+\sqrt{2}})$. However, [16] argued that it is unclear how to construct TFHE using the improved result in [39] because of their circuit construction.

[37] 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, 20, 25, 31–34, 44, 46, 52], but most of them are based on pre-quantum objects like ECDSA. There have only been two round-optimal threshold signature schemes from lattices [2, 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. Subsequent to this paper, [38] introduces two-round lattice-based threshold signature from threshold linearly linear encryption.

**$N$-out-of-$N$ TFHE.** $N$-out-of-$N$ TFHE is a special case of TFHE and falls into the category of multikey FHE [4, 19, 36, 45, 47, 48]. 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 [47] and on-the-fly MPC property [45].

**Ramp secret sharing.** Ramp secret sharing was first introduced in [10]. This scheme differentiates between the number of *reconstruction parties*, $\tau_c$, who can recover the secret, and the number of *privacy parties*, $\tau_p$, 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, 30] 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 $(\tau_c, \tau_p)$. 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 provide a concise overview of TFHE from current secret sharing schemes—Shamir secret sharing [50] and $\{0,1\}$-LSSS [14, 40]—to describe the improvements of new TFHE and UT induced by TreeSSS.

Assume a LWE based fully homomorphic encryption scheme FHE such as [17, 18, 21, 28] is given. Let $\mathsf{ct}$ 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 $\mathsf{ct}$ and $\mathsf{sk}$ 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 $\mathsf{sk}$ 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 $(c_i, \mathsf{sk}_i)$ and securely share the secret share $\mathsf{sk}_i$ 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 $\mathsf{sk}_i$ 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.

In this scenario, Shamir secret sharing scheme [50] produces a pair $(c_i, \mathsf{sk}_i)$, where the recovery coefficient $c_i$ is the Lagrange coefficient $\lambda_i \in \mathbb{Q}$. Notably, the Lagrange coefficient $\lambda_i$ turns into an integer uopn being multiplied by $N!$. Therefore, to integrate Shamir secret sharing scheme with FHE, it is necessary to multiply $N_i$ by $e_i$ to ensure $c_i \cdot (N! \cdot e_i) = (\lambda_i \cdot N!) \cdot e_i$ lies over $\mathbb{Z}_q$. This requirement indicates that $q$ must exceed $N! \cdot \lambda_i$, leading the construction that $\log q = O(N \log N)$.

On the other hand, $\{0,1\}$-LSSS [14, 40] ensures that the recovery coefficient $c_i$ is binary, thereby achieving the compactness of TFHE. Consequently, $\log q$ sets to $O(\log N)$. However, $\{0,1\}$-LSSS requires a significant number of secret shares. Specifically, for the correctness and privacy of $\{0,1\}$-LSSS, $O(N^{5.3})$ secret shares are distributed, a considerable increase compared Shamir secret sharing scheme, which distributes only $N$ secret shares.

### 2.2 Toy Example of TreeSSS: Iterative construction

This section gives a example of iterative construction of secret sharing scheme, which implies TreeSSS later section. More precisely, we consider the example

that an iteration of the Shamir secret sharing scheme can be also a secret sharing scheme.

For positive integers $N$ and $t$, $\mathsf{SS}(N,t)$ be a $t$-out-of-$N$ Shamir secret sharing scheme, and let $\mathsf{sk} \in \mathbb{Z}_q$ be a secret. We concretely provide how to construct $\mathsf{SS}(5,3)$ iteratively using $\mathsf{SS}(3,2)$.

Given a secret $\mathsf{sk}$, we splits $\mathsf{sk}$ 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 the secret $\mathsf{sk}$. 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 $\{\mathsf{sk}_j^{(3)}\}_{j\in\{1,\ldots,27\}}$ to 5 parties as follows[7]:

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

In this case, we observe that any three parties can recover the secret and any two parties can not recover the secret $\mathsf{sk}$.

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 $\mathsf{sk}$.

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 $\mathsf{sk}$. Moreover, due to the privacy property of Shamir secret sharing, $\{P_1, P_3\}$ cannot get any information about the secret $\mathsf{sk}$. This observation will be used later in the proof of the privacy properties of our TreeSSS.

### 2.3 TreeSSS: Tree Secret Sharing Scheme for $t$-out-of-$N$ Threshold Access Structure

Section 2.2 provides an example of iterative construction of Shamir secret sharing. However, we are not aware how many iterations are required to build a secret sharing for $t$-out-of-$N$ threshold access structure, and why the iteration method successes.

This section provides an overview of an iteration method for $t$-out-of-$N$ threshold structure, called a TreeSSS. In other words, we provide how to generate TreeSSS 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 any $t$-out-of-$N$ threshold access structure (see Section 4.3).

[7] 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, **??** 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.

Our construction is motivated by a majority circuit and its composition. For more clear explanation, we briefly introduce basic definitions and properties of majority circuits. The detailed discussion of majority circuits will be given in Section 3.1.

**Majority Circuits.** A majority circuit/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 $x = x_1 x_2 \dots x_N$.

Valiant [53] demonstrated that one can construct $O(N^{5.3})$-size monotone Boolean formula for computing the majority function $\mathsf{MAJ}_N$ with at least $1/2$ probability for odd $N$. (For details, see Lemma 3.4.) This probabilistic construction was directly applied to build $\{0,1\}$-LSSS for $\frac{N+1}{2}$-out-of-$N$ threshold access structure using a folklore conversion from monotone Boolean formula into a matrix [14]. The transformation is based on the insight that $\mathsf{MAJ}_N$ resembles a secret sharing for $\frac{N+1}{2}$-out-of-$N$ threshold structure if we regard $\mathsf{MAJ}_N(x) = 1$ as recovery of the secret. Similarly, when $\mathsf{MAJ}_N(x) = 0$, it can be considered as a set of participants of size less than $\frac{N}{2}$ cannot recover the secret.

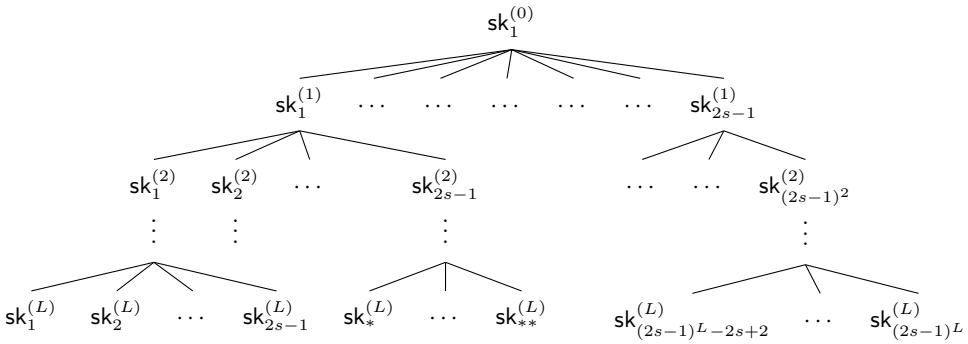
Thus, one can expect that when $\mathsf{MAJ}_N$ can be constructed with nice properties, then it might be considered as building a linear secret sharing scheme for $\frac{N+1}{2}$-out-of-$N$ threshold access structure. One of well known methodology to build $\mathsf{MAJ}_N$ is to iteratively exploit small input majority circuits $\mathsf{MAJ}_{2s-1}$, where $s \ll N$, proved by Gupta and Mahajan [37]. Unfortunately, Goldreich [35] has already argued that the folklore conversion in [14] is not effective to make $\mathsf{MAJ}_N$: One can only generates $O(N^7)$-size monotone Boolean formula for $\mathsf{MAJ}_N$, which results in a worse asymptotic size than $O(N^{5.3})$. (For details, refer to Lemma 3.5.)

**Initial Observation.** We again delve into a relation that $\mathsf{MAJ}_N$ can be built from the small input majority circuits, denote by $\mathsf{MAJ}_{2s-1}$, where $s \ll N$. We further consider $s$-out-of-$2s-1$ Shamir secret sharing, $\mathsf{SS}(2s-1, s)$ instead of $\mathsf{MAJ}_{2s-1}$. This insight is not obvious, but we deduce a relation as follows:

- We observe that the building block of Valiant's threshold circuit [53] can be represented by a small matrix, called a unit matrix. The whole circuit is also obtained by iterations of the unit matrix. (For details, refer to Appendix B.)
- Shamir secret sharing deploys the Vandermonde's *matrix*.
- We plug the Vandermonde's matrix into the iterative process of the matrices, rather than using the unit matrix.
- Indeed, we can generate $\mathsf{SS}(5,3)$ from $\mathsf{SS}(3,2)$ using 3 iterations.

The observation presents a new $t$-out-of-$N$ linear secret sharing scheme, called the TreeSSS. Fig. 3 gives a high-level overview of TreeSSS: Given the secret key $\mathsf{sk} = \mathsf{sk}_1^{(0)}$ at the root of the tree, a key dealer generates secret shares along the

tree using $s$-out-of-$2s-1$ Shamir secret sharing $\mathsf{SS}(2s-1, s)$, and then distributes leaves nodes to the participants. Working backwards from the leaf nodes of the tree to root, we can reconstruct the secret $\mathsf{sk}$.




Distribute level-$L$ secret shares $\{\mathsf{sk}_i^{(L)}\}$ to each party [8]

Fig. 3: High-level Overview of TreeSSS

As an example, we describe how TreeSSS works using 2-out-of-3 Shamir secret sharing scheme $\mathsf{SS}(3, 2)$. Let $\mathsf{sk}$ be a secret key that we want to share. For the leveled secret sharing, we regard $\mathsf{sk}$ 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 $0 \le i < L$, 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, a dealer distribute level-$L$ secret shares $\{\mathsf{sk}_j^{(L)}\}_{j\in[3^L]}$ to $N$ parties $P = \{P_1, \ldots, P_N\}$.

Here, a parameter $L$, which determines the correctness of the TreeSSS algorithm, exactly matches the results proved by [37]. Indeed, the correctness of TreeSSS is inspired by the following lemma.

**Lemma 2.1 ([37])** *Given an odd number $N$, one can construct the majority function $\mathsf{MAJ}_N$ from small majority gates $\mathsf{MAJ}_{2s-1}$ with at least 1/2 success probability. This construction requires iteratively applying $\mathsf{MAJ}_{2s-1}$-gates up to level $\log_{c_s}(N) + \log_s N + O(1)$, where $c_s = \frac{2s-1}{2^{2s-2}} \cdot \binom{2s-2}{s-1}$.*

Consequently, it is evident that the number of secret shares equals $(2s-1)^L$, which can asymptotically estimated as $O(N^{3+\frac{2.3}{\log s}})$ as in Table 1.

[8] The method for distributing level-$L$ secret shares is exactly the same as $\{0,1\}$-LSSS. Informally, the distributor randomly partitions the set $1, \ldots, (2s-1)^L$ 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.

**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.

### 2.4 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 concept of the partial decryption algorithm. The algorithm works by taking the secret shares $\mathsf{sk}_i$ and computing $\mathsf{sk} = \sum_i c_i \cdot \mathsf{sk}_i$, where $c_i$ are the recovery coefficients and $\mathsf{sk}$ 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[9]:

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

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 $\mathsf{sk}_i$. 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. We remark that in [40], the authors proposed $\{0,1\}$-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 [1,51]. The result is a new lemma which completes the security proof for the proposed construction.

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

**Lemma 2.2 (Adaptation from [14])** *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and $\mathbb{A}_{N,t}$ a $t$-out-of-$N$ threshold access structure on $P$. Consider Shamir secret sharing scheme $\mathsf{SS}$ over the secret space $\mathbb{Z}_q$, where $q$ is a prime number such that $(N!)^2 \leq q$. Then, for any set $S \subseteq [N] \cup \{0\}$ of 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.*

*Proof (of Lemma 2.2).* 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 $(N!)^2$.

## 3 Preliminaries

**Notations.** We use bold uppercase letters for matrices and bold lowercase letters for vectors. The set $[n] = 1, 2, \ldots, n$ 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 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, 47])** *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 [18, 21, 23].

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

### 3.1 Majority Circuits

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

**Definition 3.2 (Monotone Boolean formula [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 3.3 (Majority Function/Gate)** *A majority function/gate $\mathsf{MAJ}_N : \{0,1\}^N \to \{0,1\}$ is a function defined as follows:*

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

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

We now provide a summary of results that demonstrate the construction of majority functions from monotone Boolean formulas, as proven by [35, 37, 53]. However, please refer to the original papers for a full understanding and proof of these results.

**Lemma 3.4 ([53])** *Let $\gamma = 2(3-\sqrt{5}) \approx 1.52$, and $N$ be an even number of parties. If the unit circuit $\varphi = (\varphi_1 \wedge \varphi_2) \vee (\varphi_3 \wedge \varphi_4)$ is iteratively constructed with an iteration number $L \geq \log_\gamma N + \log N + O(1)$, then there exists an $O(N^{5.3})$-size monotone formula for computing the majority function $\mathsf{MAJ}_N$.*

Note that the construction presented in Lemma 3.5 involves converting the $\mathsf{MAJ}_3$ gate into a formula consisting only of AND/OR gates, which results in a larger formula compared to the construction in [53]. The conversion of the $\mathsf{MAJ}_3$ gate into AND/OR gates involves replacing $\mathsf{MAJ}_3(F_1, F_2, F_3)$ with $(F_1 \wedge F_2) \vee (F_2 \wedge F_3) \vee (F_3 \wedge F_1)$. However, this conversion is necessary as there is currently no known way to convert majority gates in circuits into matrices, unlike AND/OR gates, which can be converted via the folklore algorithm.

**Lemma 3.5 ([35])** *Let $N$ be the number of parties with odd value. Then, there exists a construction of the majority function $\mathsf{MAJ}_N$ using the small majority gate $\mathsf{MAJ}_3$. Specifically, $\mathsf{MAJ}_N$ can be constructed from $\mathsf{MAJ}_3$ 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 $\mathsf{MAJ}_N$ of size $O(N^7)$.*

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

### 3.2 Fully Homomorphic Encryption

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

**Definition 3.6 (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 $\lambda$, and a depth bound $d$, and outputs a pair of the public key $\mathsf{pk}$ and secret key $\mathsf{sk}$.*
- *The encryption algorithm $\mathsf{FHE.Enc}(\mathsf{pk}, \mu)$ takes as input $\mathsf{pk}$ and a message $\mu \in \{0,1\}$, and outputs a ciphertext $\mathsf{ct}$.*
- *The evaluation algorithm $\mathsf{FHE.Eval}(C, \mathsf{ct}_1, \ldots, \mathsf{ct}_l, \mathsf{pk})$ takes as input $l$-input circuit $C$ with less than or equal depth $d$, a bunch of ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_l$ and $\mathsf{pk}$, and outputs an evaluated ciphertext $\hat{\mathsf{ct}}$.*
- *The decryption algorithm $\mathsf{FHE.Dec}(\mathsf{pk}, \mathsf{sk}, \hat{\mathsf{ct}})$ takes as input $\mathsf{pk}$, $\mathsf{sk}$ and $\hat{\mathsf{ct}}$, and outputs a message $\mu \in \{0,1\}$.*

Hereafter, we use notations from Definition 3.6.

**Definition 3.7 (Evaluation Correctness)** *We say FHE scheme is correct if for any evaluated ciphertext $\hat{ct}$ generated by $\mathsf{FHE.Eval}(C, ct_1, \ldots, ct_l, pk)$ satisfies*

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

**Definition 3.8 (Compactness)** *We say FHE scheme is compact if for any ciphertext ct generated from the algorithm of FHE.Enc, there is a polynomial poly such that $|ct| \leq poly(\lambda, d)$.*

**Definition 3.9 (Semantic security)** *We say that FHE is secure if for all security parameter $\lambda$ 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 $(\lambda, d)$, the challenger runs $(pk, sk) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$ and $ct \leftarrow \mathsf{FHE.Enc}(pk, b)$ for $b \leftarrow \{0, 1\}$.*
2. *The challenger sends $(pk, ct)$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs a guess $b'$.*
4. *The experiment outputs 1 if $b = b'$.*

**Definition 3.10 (Special FHE)** *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 $\lambda$ and a depth bound $d$, and outputs $(pk, sk)$, where $pk$ contains a prime $q$, and $sk \in \mathbb{Z}_q^n$ for some $n = poly(\lambda, d)$.*
- *The decryption algorithm Dec consists of two functions $(\mathsf{Dec}_0, \mathsf{Dec}_1)$ defined as follows:*
  - *$p \leftarrow \mathsf{Dec}_0(sk, ct)$: $p$ 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 $\mu = \begin{cases} 0 & \text{if } p \in [-\lfloor \frac{q}{4} \rceil, \lfloor \frac{q}{4} \rceil] \\ 1 & \text{otherwise.} \end{cases}$*
- *$\mathsf{Dec}_0$ is a linear function over $\mathbb{Z}_q$ such as inner product and matrix multiplication in the secret key $sk$.*

### 3.3 Non-interactive Zero Knowledge Proof and Commitments

This section introduces the building blocks for constructing the universal thresholdizer, which are not defined in the main body of this paper. The descriptions of these schemes are based on [14].

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

**Definition 3.11 (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 $(\sigma_P, \sigma_V)$. 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 $\lambda$.*
- ***Zero knowledge**: There is a PPT simulator $S$ such that for any $(x, w) \in R$, no one can computationally distinguish two distributions:*

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

**Lemma 3.12 ([41, 49])** *PZK can be constructed from one-way functions.*

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

**Definition 3.13 (Non-interactive Commitment [11])** *We say that $\mathsf{C} = (\mathsf{C.Com})$ is a non-interactive commitment scheme if the following holds: Let $\mathsf{com}$ 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:*

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

**Lemma 3.14 ([11])** *A non-interactive commitment can be built from injective one-way functions.*

## 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 Scheme

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 $P = \{P_1, \ldots, P_N\}$ 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." We say that $S \subset P$ is a maximal invalid party if $S \notin \mathbb{A}_{N,t}$, but it holds $S \cup \{P_i\}$ for every $P_i \in P \setminus S$.*

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 $\mathsf{SS}$ 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 $r_2, \ldots, r_n \leftarrow \mathbb{Z}_q$ and generates a vector $(\mathsf{share}_1, \ldots, \mathsf{share}_d)^T = \mathbf{M} \cdot (\mathsf{sk}, r_2, \ldots, r_n)^T$. 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*

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

  *Then, $S$ can recover a secret key $\mathsf{sk}$ 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 $\mathsf{sk}$ 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 $(\mathsf{sk}_{b,1}, \ldots, \mathsf{sk}_{b,N}) \leftarrow \mathsf{SS.Share}(\mathsf{sk}_b, \mathbb{A}_{N,t})$ for $b \in \{0, 1\}$ follow the identical distribution*

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

### 4.2 TreeSSS from Shamir Secret Sharing

We propose a Tree Secret Sharing Scheme for $t$-out-of-$N$ threshold structure (TreeSSS). TreeSSS is based on a number of iterations of Shamir secret sharing scheme, and its intuition is inspired by classical results on threshold circuits from the literature [35,37]. TreeSSS allows us to construct $t$-out-of-$N$ threshold access structure from Shamir secret sharing for $s$-out-of-$\ell$ 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 Definition 4.2, TreeSSS also consists of two algorithms, called $\mathsf{TreeSSS.Share}$ and $\mathsf{TreeSSS.Combine}$.

Prior to introducing the TreeSSS, we first recall the $\mathsf{SS.Share}$. The final step of this algorithm involves generating a vector

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

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 $P_i$. To construct a share matrix $\mathbf{M}$, we deploy a core algorithm, called a TreeSS algorithm (Algorithm 1). Then, the following proposition holds. We defer its proof to Section 4.3.

**Proposition 4.5** *Let $s \geq 3$, and $\ell = 2s - 1$. Let $L \geq 1$. Then, given $L$-TreeSS algorithm (Algorithm 1) and $\mathsf{sk} \in \mathbb{Z}_q$, there exists a matrix $\mathbf{M}^{(L)} \in \mathbb{Z}_q^{\ell^L \times k_L}$ such that*

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

*where $\{\mathsf{share}_i^{(L)}\}_{i \in [\ell^L]}$ is an output of Algorithm 1, $r_i \leftarrow \mathbb{Z}_q$ for every $i$, $k_L = 1 + (s-1) \cdot \frac{\ell^L - 1}{\ell - 1} = 1 + (s-1) \cdot (1 + \ell + \ell^2 + \cdots + \ell^{L-1})$.*

For a partition $T_i \subset [\ell^L]$ associated with parties $P$, a dealer distributes $\mathsf{sk}_i = \{\mathsf{share}_j\}_{j \in T_i}$ to $P_i$ for each $i$. Consequently, $\mathsf{TreeSSS.Share}$ consists of two parts:

1. For a predefined parameter $L$ and the secret $\mathsf{sk}$, 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 $i$ to $T_k$, which means that $\mathsf{sk}_{P_k} \leftarrow \mathsf{sk}_{P_k} \cup \{\mathsf{share}_i^{(L)}\}$. After $\ell^L$ experiments, a distributor forwards the set of shares $\mathsf{sk}_k = \mathsf{sk}_{P_k} = \{\mathsf{share}_i^{(L)}\}_{i \in T_k}$ 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 $\mathsf{sk}$ as the unique level-0 secret share, denoted by $\mathsf{share}_1^{(0)}$. For each $0 \leq i \leq L$, one can generates level-$i$

**Algorithm 1:** TreeSS Algorithm

**Input :** Shamir secret sharing scheme for $s$-out-of-$\ell$ threshold access structure $\mathsf{SS}(\ell, s)$ where $s \geq 3$ and $\ell = 2s - 1$,
Secret key $\mathsf{sk}$,
Iteration number $L$

**Output:** Output the set of secret shares $\{\mathsf{share}_j\}_{j\in[\ell^L]}$.

$(\mathsf{share}_1^{(1)}, \ldots, \mathsf{share}_\ell^{(1)}) \leftarrow \mathsf{SS}(\ell, s)(\mathsf{sk})$
**for** $i = 2$ **to** $L$ **do**
  **for** $j = 1$ **to** $\ell^{i-1}$ **do**
    $(\mathsf{share}_{\ell\cdot(j-1)+1}^{(i)}, \ldots, \mathsf{share}_{\ell\cdot(j-1)+\ell}^{(i)}) \leftarrow \mathsf{SS}(\ell, s)(\mathsf{share}_j^{(i-1)})$
  **end for**
**end for**
**return** *A set of secret shares* $\{\mathsf{share}_j^{(L)}\}_{j\in[\ell^L]}$.

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, $\{\mathsf{share}_k^{(i)}\}_{k\in\{\ell\cdot(j-1)+1,\ldots,\ell\cdot j\}}$. 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 $\mathsf{TreeSSS.Share}$ is provided in Algorithm 2.

$\mathsf{TreeSSS.Combine}$ 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 $\mathsf{sk}$. The full algorithm is given by Algorithm 3. Here, we note that the existence of recovery coefficients $c_j^S$ such that $\mathsf{sk} = \sum_{j\in\bigcup_{P_i\in S} T_i} c_j^S \cdot \mathsf{share}_j^{(L)}$ as in Definition 4.2. In addition, $c_j^S$ is a product of Lagrange coefficients of the form $\lambda_{j_1}^{S_{j_1}^{(1)}} \cdots \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \lambda_{j_L}^{S_{j_L}^{(L)}}$ for some proper $j_i$ and $S_{j_i}^{(i)}$.

**Algorithm 2:** $\mathsf{TreeSSS.Share}$

**Input :** Parties $P = \{P_1, \ldots, P_N\}$ with odd $N$, secret key $\mathsf{sk}$,
$\frac{N+1}{2}$-ouf-of-$N$ threshold access structure $\mathbb{A}_{N,\frac{N+1}{2}}$

**Output:** Output the set of secret shares $\{\mathsf{share}_j\}_{j\in[\ell^L]}$.

Choose $s \geq 3$ and $\ell = 2s - 1$, and calculate the iteration number $L$ such that $L \geq \log_{c_s} N + \log_s N + O(1)$ where $c_s = \frac{\ell}{2^{\ell-1}} \cdot \binom{\ell-1}{s-1}$
Generate a partition $T_i \subset [\ell^L]$ associated with parties $P = \{P_1, \ldots, P_N\}$
**for** $i = 1$ **to** $\ell^L$ **do**
  Sample $k \leftarrow [N]$, and $i \in T_k$
**end for**
Compute $\{\mathsf{share}_i^{(L)}\}_{i\in[\ell^L]} \leftarrow L\text{-TreeSS}(\mathsf{SS}(\ell, s), \mathsf{sk})$ (Algorithm 1).
Define $\mathsf{sk}_i = \{\mathsf{share}_j^{(L)}\}_{j\in T_i}$ and a dealer sends it to $P_i$
**return** *A set of secret shares* $\mathsf{sk}_i$

**Algorithm 3:** TreeSSS.Combine

**Input :** $B$: a set of secret shares $\{\mathsf{share}_j^{(L)}\}_{j \in \bigcup_{P_i \in S} T_i}$ where $S$ is a subset of $P = \{P_1 \ldots, P_N\}$.

**Output:** Recovered secret key $\hat{\mathsf{sk}}$.

**if** $S \notin \mathbb{A}_{N,\frac{N+1}{2}}$ **then**
  **return** $\perp$
**end if**
**for** $k = L$ **downto** $0$ **do**
  Compute the following sets:

  $T^{(k)} = \{j_k \mid \mathsf{share}_{j_k}^{(k)}$ can be recovered by $B\}$

  $S_{j_k}^{(k)} = \{\mathsf{share}_{j_k}^{(k)} \mid \mathsf{share}_{j_k}^{(k)}$ recovers $\mathsf{share}_{j_{k-1}}^{(k-1)}$ for every $j_{k-1} \in T^{(k-1)}\}$

  Compute the Lagrange coefficients $\lambda_{j_k,0}^{S_{j_k}^{(k)}} = \lambda_{j_k}^{S_{j_k}^{(k)}}$ of $\mathsf{SS}(\ell, s)$
**end for**
**if** $T^{(0)} = \emptyset$ **then**
  **return** $\perp$
**end if**
Compute $\hat{\mathsf{sk}} = \sum_{j_L \in T^{(L)}} \lambda_{j_1}^{S_{j_1}^{(1)}} \cdots \lambda_{j_{L-1}}^{S_{j_{L-1}}^{(L-1)}} \cdot \lambda_{j_L}^{S_{j_L}^{(L)}} \cdot \mathsf{share}_{j_L}^{(L)}$
**return** $\hat{sk}$.

In summary, we obtain the following theorem.

**Theorem 4.6** *Let $N$ be an odd number and $P = \{P_1, \ldots, P_N\}$ be a set of parties. Let $s$ be a small positive integer such that $s \ll N$, and $\ell = 2s - 1$. Given Shamir 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 for $\frac{N+1}{2}$-out-of-$N$ threshold structure satisfies the correctness (Definition 4.3) and the privacy (Definition 4.4) with at least 1/2 success probability and the number of secret shares is $\ell^L = O(N^{\log_{c_s} \ell + \log_s \ell})$.*

### 4.3 Proofs of TreeSSS

This section provides proofs of TreeSSS, especially Proposition 4.5 and Theorem 4.6. We first remark that $\hat{\mathsf{sk}}$ equals to $\mathsf{sk}$, input of Algorithm 2. By definition of $T^{(k)}, S_{j_k}^{(k)}$ and $\lambda_{j_k}^{S_{j_k}^{(k)}}$, we easily verify the following equations when level-$k$ sets, and Lagrange coefficients are well computed for every $k$. More precisely, we can get following equation for every $k$ and $j_{k-1} \in T^{(k-1)}$ by using Lagrange coefficients.

$$\mathsf{share}_{j_{k-1}}^{(k-1)} = \sum_{j_k \in S_{j_k}^{(k)}} \lambda_{j_k}^{S_{j_k}^{(k)}} \cdot \mathsf{share}_{j_k}^{(k)}.$$

Now, we show that $\hat{\mathsf{sk}}$ can be sequentially calculated to result in $\mathsf{share}_1^{(0)}$, which is same as our target secret key $\mathsf{sk}$.

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

For the correctness, we need to show that the 0-level share set $T^{(0)} = \{\mathsf{sk}\}$, which can be recovered by $B$, is not empty.

Then, in order to show that $L$-TreeSSS is a linear secret sharing scheme that satisfies Proposition 4.5 and Theorem 4.6, it suffices to prove that

- The existence of a share matrix $\mathbf{M}^{(L)}$ that satisfies Definition 4.2.
- Correctness of TreeSSS (Definition 4.3)
- Privacy of TreeSSS (Definition 4.4).

We now show the existence of a share matrix $\mathbf{M}^{(L)}$ as in Proposition 4.5.

*Proof (of Proposition 4.5).* We use the mathematical induction on $L$. For $L = 1$, it is obvious since we just take the Vandermonde's matrix $\mathbf{V}_s = \mathbf{M}^{(1)}$. Suppose that there exists a matrix $\mathbf{M}^{(L-1)} \in \mathbb{Z}_q^{\ell^{L-1} \times k_{L-1}}$ such that

$$(\mathsf{share}_1^{(L-1)}, \ldots, \mathsf{share}_{\ell^{L-1}}^{(L-1)})^T = \mathbf{M}^{(L-1)} \cdot (\mathsf{sk}, r_2, \ldots, r_{k_{L-1}})^T,$$

where $\{\mathsf{share}_i^{(L-1)}\}$ is a set of secret shares obtained by $(L-1)$-TreeSS algorithm.

By Algorithm 1, level-$L$ secret shares $\mathsf{share}_i^{(L)}$ can be obtained as follows: For each $j$, when a dealer conducts $\mathsf{SS}(\ell, s)(\mathsf{share}_j^{(L-1)})$, then $\ell$ level-$L$ secret shares $\{\mathsf{share}_{\ell(j-1)+1}^{(L)}, \ldots, \mathsf{share}_{\ell \cdot j}^{(L)}\}$ are obtained. By definition of $\mathsf{SS}(\ell, s)$, we observe that for every $j$, it satisfies that

$$(\mathsf{share}_{\ell(j-1)+1}^{(L)}, \ldots, \mathsf{share}_{\ell \cdot j}^{(L)})^T = \mathbf{V}_s \cdot (\mathsf{share}_j^{(L-1)}, r_{2,j}^{(L-1)}, r_{3,j}^{(L-1)}, \ldots, r_{s,j}^{(L-1)})^T.$$

For simple description, we define some notations

$$\mathbf{r}_j^{(L-1)} = (r_{2,j}^{(L-1)}, r_{3,j}^{(L-1)}, \ldots, r_{s,j}^{(L-1)})^T \in \mathbb{Z}_q^{s-1}$$

$$
\begin{aligned}
\mathbf{r}^{(L-1)} &= ((\mathbf{r}_1^{(L-1)})^T, (\mathbf{r}_2^{(L-1)})^T, \ldots, (\mathbf{r}_{\ell^{L-1}}^{(L-1)})^T)^T \in \mathbb{Z}_q^{(s-1)\cdot\ell^{L-1}} \\
\mathbf{s}_j^{(L-1)} &= (\mathsf{share}_j^{(L-1)}, \mathbf{r}_j^{(L-1)}) \in \mathbb{Z}_q^s \\
\overline{\mathsf{share}}_j^{(L)} &= (\mathsf{share}_{\ell(j-1)+1}^{(L)}, \ldots, \mathsf{share}_{\ell\cdot j}^{(L)})^T \in \mathbb{Z}_q^\ell
\end{aligned}
$$

for every $L > 1$ and $j$. We thus write the above equation by

$$
\overline{\mathsf{share}}_j^{(L)} = \mathbf{V}_s \cdot \mathbf{s}_j^{(L-1)}
$$

for every $L$ and $j$. Hence, it holds that

$$
\begin{pmatrix}
\overline{\mathsf{share}}_1^{(L)} \\
\overline{\mathsf{share}}_2^{(L)} \\
\overline{\mathsf{share}}_3^{(L)} \\
\vdots \\
\overline{\mathsf{share}}_{\ell^{L-1}}^{(L)}
\end{pmatrix}
=
\begin{pmatrix}
\mathbf{V}_s & & & & \\
& \mathbf{V}_s & & & \\
& & \mathbf{V}_s & & \\
& & & \ddots & \\
& & & & \mathbf{V}_s
\end{pmatrix}
\cdot
\begin{pmatrix}
\mathbf{s}_1^{(L-1)} \\
\mathbf{s}_2^{(L-1)} \\
\mathbf{s}_3^{(L-1)} \\
\vdots \\
\mathbf{s}_{\ell^{L-1}}^{(L-1)}
\end{pmatrix}
$$

Then, there exists a permutation matrix $\mathbf{P}$ such that

$$
\begin{aligned}
\begin{pmatrix}
\overline{\mathsf{share}}_1^{(L)} \\
\overline{\mathsf{share}}_2^{(L)} \\
\overline{\mathsf{share}}_3^{(L)} \\
\vdots \\
\overline{\mathsf{share}}_{\ell^{L-1}}^{(L)}
\end{pmatrix}
&=
\begin{pmatrix}
\mathbf{V}_s & & & & \\
& \mathbf{V}_s & & & \\
& & \mathbf{V}_s & & \\
& & & \ddots & \\
& & & & \mathbf{V}_s
\end{pmatrix}
\cdot \mathbf{P}^{-1} \cdot \mathbf{P}
\begin{pmatrix}
\mathbf{s}_1^{(L-1)} \\
\mathbf{s}_2^{(L-1)} \\
\mathbf{s}_3^{(L-1)} \\
\vdots \\
\mathbf{s}_{\ell^{L-1}}^{(L-1)}
\end{pmatrix} \\
&=
\begin{pmatrix}
\mathbf{V}_s & & & & \\
& \mathbf{V}_s & & & \\
& & \mathbf{V}_s & & \\
& & & \ddots & \\
& & & & \mathbf{V}_s
\end{pmatrix}
\cdot \mathbf{P}^{-1} \cdot
\begin{pmatrix}
\mathsf{share}_1^{(L-1)} \\
\mathsf{share}_2^{(L-1)} \\
\vdots \\
\mathsf{share}_{\ell^{L-1}}^{(L-1)} \\
\mathbf{r}^{(L-1)}
\end{pmatrix}
\end{aligned}
$$

where $\mathbf{r}^{(L-1)}$ is a vertical concatenation of $\mathbf{r}_j^{(L-1)}$ for every $j$. By the mathematical hypothesis, it satisfies that

$$
\begin{pmatrix}
\mathsf{share}_1^{(L-1)} \\
\mathsf{share}_2^{(L-1)} \\
\vdots \\
\mathsf{share}_{\ell^{L-1}}^{(L-1)}
\end{pmatrix}
= \mathbf{M}^{(L-1)} \cdot
\begin{pmatrix}
\mathsf{sk} \\
r_2 \\
\vdots \\
r_{k_{L-1}}
\end{pmatrix}.
$$

Thus, it holds that

$$\begin{pmatrix} \mathsf{share}_1^{(L-1)} \\ \mathsf{share}_2^{(L-1)} \\ \vdots \\ \mathsf{share}_{\ell^{L-1}}^{(L-1)} \\ \mathbf{r}^{(L-1)} \end{pmatrix} = \begin{pmatrix} \mathbf{M}^{(L-1)} & \\ & \mathbf{I} \end{pmatrix} \cdot \begin{pmatrix} \mathsf{sk} \\ r_2 \\ \vdots \\ r_{k_{L-1}} \\ \mathbf{r}^{(L-1)} \end{pmatrix}.$$

Finally, we have

$$\begin{aligned} \begin{pmatrix} \mathsf{share}_1^{(L)} \\ \mathsf{share}_2^{(L)} \\ \mathsf{share}_3^{(L)} \\ \vdots \\ \mathsf{share}_{\ell^L}^{(L)} \end{pmatrix} &= \begin{pmatrix} \overline{\mathsf{share}}_1^{(L)} \\ \overline{\mathsf{share}}_2^{(L)} \\ \overline{\mathsf{share}}_3^{(L)} \\ \vdots \\ \overline{\mathsf{share}}_{\ell^{L-1}}^{(L)} \end{pmatrix} \\ &= \underbrace{\begin{pmatrix} \mathbf{V}_s & & & & \\ & \mathbf{V}_s & & & \\ & & \mathbf{V}_s & & \\ & & & \ddots & \\ & & & & \mathbf{V}_s \end{pmatrix} \cdot \mathbf{P}^{-1} \cdot \begin{pmatrix} \mathbf{M}^{(L-1)} & \\ & \mathbf{I} \end{pmatrix}}_{\mathbf{M}^{(L)}} \cdot \begin{pmatrix} \mathsf{sk} \\ r_2 \\ \vdots \\ r_{k_{L-1}} \\ \mathbf{r}^{(L-1)} \end{pmatrix}. \end{aligned}$$

Then we compute the number of input value $\{\mathsf{sk}, r_1, \ldots, r_{k_{L-1}}, (\mathbf{r}^{(L-1)})^T\}$.

$$\begin{aligned} k_{L-1} + (s-1) \cdot \ell^{L-1} &= 1 + (s-1)\frac{\ell^{L-1} - 1}{\ell - 1} + (s-1) \cdot \ell^{L-1} \\ &= 1 + (s-1)\frac{\ell^{L} - 1}{\ell - 1} \\ &= k_L. \end{aligned}$$

We easily confirm that $\mathbf{M}^{(L)} \in \mathbb{Z}_q^{\ell^L \times k_L}$. By the mathematical induction, we complete the proof. □

*Remark 2.* The proof of Proposition 4.5 does not give the closed form of $\mathbf{M}^{(L)}$. However, since it is a constructive proof of $\mathbf{M}^{(L)}$, the output of Algorithm 1 can be regarded as a matrix.

Motivated by Lemma 2.1, 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.

*Proof (of Theorem 4.6).* Let $L \geq \log_{c_s} N + \log_s N + O(1)$ where $c_s = \frac{\ell}{2^{\ell-1}} \cdot \binom{\ell-1}{s-1}$. Then, by the size of $\mathbf{M}^{(L)}$, we easily confirm that $\ell^L = O(N^{\log_{c_s} \ell + \log_s \ell})$ is the total number of secret shares, which is approximated to $O(N^{3+2.3/\log s})$.[10]

[10] The detailed computation of approximations will be given by Appendix A.

**Correctness.** In order to prove the correctness of $L$-TreeSSS, it suffices to show that the probability of $T^{(0)} \neq \emptyset$ is at least $1/2$. Before proceeding with the proof, we introduce two lemmas.

**Lemma 4.7** *Let $B_S$ be a set of secret shares $\{\mathsf{share}_j^{(L)}\}_{j \in \bigcup_{P_i \in S} T_i}$ for some subset $S \subset P$ and $L \geq log_{c_s} N + log_s N + O(1)$. For any $S \subset P$ of size $\frac{N+1}{2}$ or more, it holds that*

$$\Pr[\perp \leftarrow \mathsf{TreeSSS.Combine}(B_S)] < \frac{1}{2^{N+1}}. \tag{1}$$

*In addition, for any $S' \subset P$ of size less than $\frac{N+1}{2}$, it satisfies that*

$$\Pr[\hat{sk} \leftarrow \mathsf{TreeSSS.Combine}(B_{S'})] < \frac{1}{2^{N+1}}. \tag{2}$$

*This lemma implies that all subset of $P$ satisfy the correctness of the* TreeSSS *with a probability of at least $1 - \frac{1}{2^{N+1}}$.*

According to this Lemma 4.7, we can compute the lower bound of the probability that TreeSSS satisfies the correctness as follows:

$$\begin{aligned}
\Pr[\text{Correctness}] &\geq 1 - \sum_{S \subset P, |S| \geq \frac{N+1}{2}} \Pr[\perp \leftarrow \mathsf{TreeSSS.Combine}(B_S)] \\
&\quad - \sum_{S' \subset P, |S'| < \frac{N+1}{2}} \Pr[\hat{\mathsf{sk}} \leftarrow \mathsf{TreeSSS.Combine}(B_{S'})] \\
&\geq 1 - 2^N \cdot \frac{1}{2^{N+1}} = 1/2.
\end{aligned}$$

Before proving Lemma 4.7, we introduce one more lemma.

**Lemma 4.8 ([37])** *Let $X_1, \ldots, X_{2s-1} \leftarrow \{0,1\}$ be independent identically distributed random variables, and $p := \Pr[X_i = 1]$ for all $i$ and $s \geq 2$. Then, following properties hold:*

1. $p' := \Pr[\sum_{i=1}^{2s-1} X_i \geq s] = \sum_{j=0}^{s-1} \binom{2s-1}{j} \cdot p^{2s-1-j}(1-p)^j$.
2. $p' \geq p$ *when* $p \geq 0.5$, $p' \leq p$ *when* $p \leq 0.5$
3. $\delta := p - 0.5$, *it holds that* $p' = 0.5 + (c_s - O(\delta)) \cdot \delta$ *where* $c_s = \frac{\ell}{2^{\ell-1}} \cdot \binom{\ell-1}{s-1}$.
4. $1 - p' < 8^{s-1} \cdot (1-p)^s$.
5. $p' < 8^{s-1} \cdot p^s$.

*Proof (of Lemma 4.8).* The proof of each item is straightforward.

1. By definition of $p'$, we directly calculate the following

$$\begin{aligned}
p' &= \Pr[\sum_{i=1}^{2s-1} X_i \geq s] \\
&= \sum_{j=0}^{s-1} \binom{2s-1}{j} \cdot p^{2s-1-j}(1-p)^j.
\end{aligned}$$

2. To prove inequalities, we define a function $f$ and differentiate it as follows:

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

   We know that $f(0) = f(0.5) = f(1) = 0$. When $1 \geq p \geq 0.5$, $f''(p) \leq 0$. Then, $f$ is a concave function so that $f(p) \geq 0$ if $1 \geq p \geq 0.5$. Similarly, When $0.5 \geq p \geq 0$, $f''(p) \geq 0$. Then, $f$ is a convex function so that $f(p) \leq 0$ if $0.5 \geq p \geq 0$.
3. Since $p = 0.5 + \delta$, it holds that

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

4. By the basic combinatorics, it holds that

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

$$< 4^{s-1} \cdot 2 \cdot (1-p)^s = 2^{2s-1} \cdot (1-p)^s$$
$$\leq 2^{3s-3} \cdot (1-p)^s = 8^{s-1} \cdot (1-p)^s.$$

5. By the basic combinatorics, it holds that

$$\begin{aligned} p' &= \sum_{j=0}^{s-1} \binom{2s-1}{j} \cdot p^{2s-1-j}(1-p)^j \\ &< \binom{2s-1}{s-1} p^s \cdot \sum_{j=0}^{s-1} \binom{s-1}{j} \cdot p^{s-1-j}(1-p)^j \\ &= \binom{2s-1}{s-1} p^s \\ &< 2^{2s-1} \cdot p^s \leq 2^{3s-3} \cdot p^s = 8^{s-1} \cdot p^s. \end{aligned}$$

□

We are ready to prove Lemma 4.7 to finish the proof of the correctness.

*Proof (of Lemma 4.7).* Without loss of generality, it suffices to prove Eq. (1), especially $|S| = \frac{N+1}{2}$. In other words, given $B_S$ and $L$ as above, we will prove that

$$\Pr[\perp \leftarrow \mathsf{TreeSSS.Combine}(B_S)] < \frac{1}{2^{N+1}}.$$

By construction of $\mathsf{TreeSSS.Share}$, a family of level-$L$ secret shares $\{\mathsf{share}_j^{(L)}\}_{j \in [\ell^L]}$ is uniformly distributed to each party. Thus, the probability that $S$ has a specific secret share $\mathsf{share}_j^{(L)}$ is easily computed by

$$\Pr[S \text{ has } \mathsf{share}_j^{(L)}] = \frac{N+1}{2N} = \frac{1}{2} + \frac{1}{2N} \text{ for every } j \in [\ell^L].$$

Let $p_{i,j}$ be the probability that $S$ has a secret share $\mathsf{share}_j^{(i)}$ for every $j \in [\ell^i]$ and $1 \leq i \leq L$. Then, it obviously holds that $p_{L,j} = \frac{1}{2} + \frac{1}{2N}$ for every $j$. We now compute $p_{i,j}$ for every $i$ and $j$. By construction of TreeSS algorithm (Algorithm 1), a secret share of the form $\mathsf{share}_{\ell\cdot(j-1)+k}^{(i)}$ with $k \in [\ell]$ is obtained by $\mathsf{SS}(\ell, s)(\mathsf{share}_j^{(i-1)})$ for every $i, j$. In other words, when $S$ has a proper set of level-$i$ secret shares, then $S$ can reconstruct a level-$(i-1)$ $\mathsf{share}_j^{(i-1)}$ by the correctness of $\mathsf{SS}(\ell, s)$.

On the other hand, reconstructing $\mathsf{share}_j^{(i-1)}$ from $s$ level-$i$ secret shares can be regarded as $\sum_i X_i \geq s$ defining $X_k$ as follows.

$$X_k = \begin{cases} 1 & \text{if } \mathsf{share}_{\ell\cdot(j-1)+k}^{(i)} \in S \\ 0 & \text{otherwise.} \end{cases}$$

for $k \in [\ell]$. The privacy of $\mathsf{SS}(\ell, s)$ guarantees that $X_k$'s are identical and independent. Thus, we directly apply Lemma 4.8 to compute $p_{i-1,j}$. Fortunately, for

every $i$, we easily observe that $p_{i,j_i} = p_{i,j'_i}$ even if $j_i \neq j'_i$ since we already know $p_{L,j_L} = \frac{1}{2} + \frac{1}{2N}$ for every index $j_L$. By combining the above equation, it implies $p_{L-1,j_{L-1}} = p_{L-1,j'_{L-1}}$ for every $j_{L-1}, j'_{L-1}$. We can therefore simplify $p_{i,j}$ by $p_j$, regardless of an index $j$. That is, we obtain

$$p_{i-1} = \sum_{k=0}^{s-1} \binom{2s-1}{k} \cdot p_i^{2s-1-k}(1-p_i)^k.$$

Furthermore, by adapting Lemma 4.8 again, we also have

$$p_{i-1} = 0.5 + (c_s - O(\delta_i)) \cdot \delta_i,$$

where $c_s = \frac{\ell}{2^{\ell-1}} \cdot \binom{\ell-1}{s-1}$ and $\delta_i = p_i - 0.5$. From this observation, we have

$$\begin{aligned} p_L &= \frac{1}{2} + \frac{1}{2N} \\ p_{L-1} &= \frac{1}{2} + \left(c_s - O\left(\frac{1}{2N}\right)\right) \cdot \frac{1}{2N} \\ p_{L-i} &= \frac{1}{2} + \left(\prod_{j=0}^{i-1} \left(c_s - O\left(\delta_{L-j}\right)\right)\right) \cdot \frac{1}{2N}. \end{aligned}$$

According to the first property of Lemma 4.8, we can compute $p_{i-1}$ as follows:

$$p_{i-1} = \sum_{j=0}^{s-1} \binom{2s-1}{j} \cdot p_i^{2s-1-j}(1-p_i)^j.$$

By the third property of Lemma 4.8, $p_{i-1}$ also holds the following equation:

$$p_{i-1} = 0.5 + (c_s - O(\delta_i)) \cdot \delta_i,$$

where $c_s = \frac{\ell}{2^{\ell-1}} \cdot \binom{\ell-1}{s-1}$ and $\delta_i = p_i - 0.5$. So, we can amplify the probability from $\frac{1}{2} + \frac{1}{2N}$ to 15/16. To compute the number of levels, we provide the lower bound of $p_{L-i}$ as follows:

$$\begin{aligned} p_L &= \frac{1}{2} + \frac{1}{2N} \\ p_{L-1} &= \frac{1}{2} + \left(c_s - O\left(\frac{1}{2N}\right)\right) \cdot \frac{1}{2N} \\ p_{L-i} &= \frac{1}{2} + \left(\prod_{j=0}^{i-1} \left(c_s - O\left(\delta_{L-j}\right)\right)\right) \cdot \frac{1}{2N} \end{aligned}$$

If $p_{L-i}$ is lower than $\frac{15}{16}$, then all $\{\delta_{L-j}\}_{0 \leq j \leq i-1}$ is less than $\frac{7}{16}$ by the second property of Lemma 4.8. Therefore, $p_{L-i} \geq \frac{1}{2} + (c_s - O(1))^i \cdot \frac{1}{2N}$ until $p_{L-i} \leq \frac{15}{16}$. Then, we can compute the lower bound of levels $i$ to exceed $\frac{15}{16}$ as follows:

$$\frac{1}{2} + (c_s - O(1))^i \cdot \frac{1}{2N} \geq \frac{15}{16}$$

$$(c_s - O(1))^i \geq \frac{7N}{8}$$
$$i \geq \log_{c_s - O(1)} \left(\frac{7N}{8}\right) \approx \log_{c_s} N + O(1)$$

Now, we will amplify the probability from $\frac{15}{16}$ to $1 - \frac{1}{2^{N+1}}$. Let $p_{L-i}$ be $\frac{15}{16}$. By the fourth property of Lemma 4.8, we can get inequalities as follows:

$$8(1 - p_{L-i-j}) \leq (8(1 - p_{L-i-j+1}))^s \leq \ldots \leq (8(1 - p_{L-i}))^{s^j} \leq \left(\frac{1}{2}\right)^{s^j}.$$

Since $1 - p_{L-i-j} \leq \frac{1}{2^{s^j+3}}$, we can compute the lower bound of levels $j$ to exceed $1 - \frac{1}{2^{N+1}}$ as follows:

$$s^j + 3 \geq N + 1$$
$$j \geq \log_s(N-2) \approx \log_s N + O(1).$$

Since $L \geq \log_{c_s} N + \log_s N + O(1)$, $p_0 \geq 1 - \frac{1}{2^{N+1}}$. Therefore, Eq. (1) holds. Similarly, Eq. (2) also holds.

**Privacy.** Now, we demonstrate that the privacy holds. 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 $\mathsf{TreeSSS.share}(\mathsf{sk}_b, \mathsf{SS}(\ell, s), L) \to (\mathsf{sk}_{b,1}, \ldots, \mathsf{sk}_{b,N})$ for $b \in 0, 1$:

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

We will show that these two pairs of shares are drawn from the same distribution. To this end, we use a mathematical induction on the level $L$. For the base case of $L = 1$, TreeSSS is equivalent to Shamir 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 when $L = 1$.

To continue the proof, we assume that a $(k+1)$-level TreeSSS, with each subtree corresponding to a $k$-level TreeSSS, satisfies the privacy. Then, it suffices to demonstrate that privacy also holds on $(k+1)$-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 will prove that $S_0^{(k+1)}$ and $S_1^{(k+1)}$ are indistinguishable. Moreover, due to iterative constructions, the level-$k$ secret shares $S_0^{(k)}, S_1^{(k)}$ can be viewed as the output of $k$-TreeSSS. Thus, $S_0^{(k)}$ and $S_1^{(k)}$ have the identical distributions because of the induction hypothesis.

Now, we divide $S_b^{(k+1)}$ 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.

By definition, a set of level-$(k+1)$ secret shares $S_{b,P}^{(k+1)}$ is derived from level-$k$ secret shares in $S_b^{(k)}$ by using Shamir secret sharing for every $b$. Since nobody can distinguish between $S_0^{(k)}, S_1^{(k)}$, it obviously holds $S_{0,P}^{(k+1)}, S_{1,P}^{(k+1)}$ also follow the identical distribution.

Moreover, $S_{0,I}^{(k+1)}, S_{1,I}^{(k+1)}$ both come from Shamir secret sharing and cannot recover level-$k$ secret shares. Therefore they also follow the identical distribution because Shamir 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 conclude that $L$-TreeSSS satisfies the privacy for all positive integer $L$.

□

### 4.4 TreeSSS for $t$-out-of-$N$ for arbitrary $t$

Theorem 4.6 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 $\mathsf{TreeSSS}(N,t)$ and prove its correctness and privacy. 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** $N$**.**

- $\mathsf{TreeSSS.Share}(\mathsf{sk}, \mathbb{A}_{N,t})$ :
  1. Sample $\{\mathsf{share}_i\}_i \leftarrow \mathsf{TreeSSS}(2t-1,t)$ by assuming the existence of ephemeral $2t-1$ parties. Suppose that each ephemeral party obtains a set of shares $S_k = \{\mathsf{share}_i\}$ for $k \in [2t-1]$.
  2. A dealer distributes $S_k$ to each $P_i$ for every $k \in [N]$.
- $\mathsf{TreeSSS.Combine}(B)$: Run a combine algorithm of $\mathsf{TreeSSS}(2t-1,t)$ given a set of shares $B$.

The correctness is evident because of $2t-1 > N$. Furthermore, the privacy of $\mathsf{TreeSSS}(N,t)$ is directly derived from that of $\mathsf{TreeSSS}(2t-1,t)$.

More precisely, the following holds:

**Correctness.** The correctness of $\mathsf{TreeSSS}(2t-1,t)$ ensures that for any set $S \subset \{P_1, \ldots, P_{2t-1}\}$ of size $t$ or more, $\mathsf{sk}$ can be recovered among parties in $S$. Otherwise, $\mathsf{sk}$ can not be recovered.
Accordingly, for any $S' \subset \{P_1, \ldots, P_N\}$ of size $t$ or more, $\mathsf{sk}$ can be also recovered among parties in $S'$. Therefore, $\mathsf{TreeSSS}(N,t)$ satisfies the correctness.

**Privacy.** The privacy of $\mathsf{TreeSSS}(2t-1,t)$ ensures that for any subset $S \subset \{P_1,\ldots,P_{2t-1}\}$ of size less than $t$, and for any secret key $\mathsf{sk}_0, \mathsf{sk}_1$ in key space, two family of shares follow the identical distribution

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

where $\mathsf{sk}_{k,i}$ is a secret share of $\mathsf{sk}_i$. Since $\{P_1,\ldots,P_N\} \subset \{P_1,\ldots,P_{2t-1}\}$, for any subset $S' \subset \{P_1,\ldots,P_N\}$ of size less than $t$, $S'$ automatically satisfies the privacy. Therefore, $\mathsf{TreeSSS}(N,t)$ satisfies the privacy, directly.

**Case 2) $\mathsf{TreeSSS}(N,t)$ for $t < \frac{N+1}{2}$ and odd $N$.**

- $\mathsf{TreeSSS.Share}(\mathsf{sk}, \mathbb{A}_{N,t})$ :
  1. Sample $\{\mathsf{share}_i\} \leftarrow \mathsf{TreeSSS}(N+r, t+r = \frac{N+r+1}{2})$ by assuming the existence of ephemeral $N+r$ parties, where $r = N-2t+1$. Suppose each ephemeral party gets a set of shares $S_k = \{\mathsf{share}_i\}$ for $k \in [N+r]$.
  2. A dealer distributes $S_k$ to each $P_i$ for every $k \in [N]$ and publicly broadcasts the remaining $r$ subset of shares $\{S_{N+1},\ldots,S_{N+r}\}$.
- $\mathsf{TreeSSS.Combine}(B)$: Given $B' = B \cup \{S_{N+1},\ldots,S_{N+r}\}$, run a combine algorithm of $\mathsf{TreeSSS}(N+r,t+r)$.

Since $B'$ can be regarded as a collection of $t+r$ parties, so the correctness is guaranteed by that of $\mathsf{TreeSSS}(N+r,t+r)$. Similarly, the privacy also holds because of that of $\mathsf{TreeSSS}(N+r,t+r)$.

More precisely, the following holds:

**Correctness.** The correctness of $\mathsf{TreeSSS}(N+r,t+r)$ ensures that for any subset $S \subset \{P_1,\ldots,P_{N+r}\}$ of size of $t+r$ or more, $\mathsf{sk}$ can be recovered. Otherwise, $\mathsf{sk}$ can not be recovered.
Moreover, for any $S' \subset \{P_1,\ldots,P_N\}$ of size $t$ or more, each party in $S'$ additionally knows $\{S_{N+1},\ldots,S_{N+r}\}$ because they are public information. Thus, if $|S'| \geq t$, the collective information held by parties can be represented as $S' \cup \bigcup\{P_{N+1},\ldots,P_{N+r}\}$. This means that the total number of involved parties could be $t+r$ or more. Consequently, $S'$ can recover $\mathsf{sk}$ because of the correctness of $\mathsf{TreeSSS}(N+r,t+r)$.

**Privacy.** The privacy of $\mathsf{TreeSSS}(N+r,t+r)$ ensures that for any subset $S \subset \{P_1,\ldots,P_{N+r}\}$ of size less than $t$, and for any secret key $\mathsf{sk}_0, \mathsf{sk}_1$ in key space, two family of shares follow the identical distribution

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

where $\mathsf{sk}_{k,i}$ is a secret share of $\mathsf{sk}_i$. Since $\{P_1,\ldots,P_N\} \subset \{P_1,\ldots,P_{N+r}\}$, for any subset $S' \subset \{P_1,\ldots,P_N\}$ of size less than $t$, $|\tilde{S} = S' \bigcup \{S_{N+1},\ldots,S_{N+r}\}|$ is still smaller than $t+r$. Therefore, if one can break the privacy of $\mathsf{TreeSSS}(N,t)$, then $\mathsf{TreeSSS}(N+r,t+r)$ is also broken, which contradicts to the assumption. Therefore, $\mathsf{TreeSSS}(N,t)$ satisfies the privacy.

**Case 3)** $\mathsf{TreeSSS}(N,t)$ **for any** $t$ **and** $N$ **is even.**

- $\mathsf{TreeSSS.Share}(\mathsf{sk}, \mathbb{A}_{N,t})$ :
  1. Sample $\{\mathsf{share}_i\} \leftarrow \mathsf{TreeSSS}(N+1, t+1)$ by assuming the existence of ephemeral $N+1$ parties. Suppose each ephemeral party gets a set of shares $S_k = \{\mathsf{share}_i\}$ for $k \in [N+1]$.
  2. A dealer distributes $S_k$ to each $P_i$ for every $k \in [N]$ and publicly broadcasts the unique remaining set $S_{N+1}$.
- $\mathsf{TreeSSS.Combine}(B)$: Given $B' = B \cup S_{N+1}$, run a combine algorithm of $\mathsf{TreeSSS}(N+1, t+1)$.

In this case, $\mathsf{TreeSSS}(N,t)$ satisfies the correctness and privacy in the similar way as Case 2).

*Remark 3.* For each case, the total number of parties is less than $2N$. Therefore, the number of secret shares is still $O(N^{\log_{cs} \ell + \log_s \ell})$ 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.

### 4.5 (Exponential) Offline cost: Concealed probability in $\{0,1\}$-LSSS

$\{0,1\}$-LSSS described in [14, 40] depends on the probabilistic construction of monotone Boolean formulas proposed by Valiant [53]. Lemma 3.4 (for $\{0,1\}$-LSSS) and Theorem 4.6 (for TreeSSS) argue that that the probability of constructing the circuits is at least $1/2$.

This only ensures that we can construct $\{0,1\}$-LSSS and TreeSSS with at least $1/2$ probability depending on the distribution of secret shares. Therefore, a dealer should check whether a share matrix $\mathbf{M}$ is correctly generated. If so, the dealer generates secret shares using $\mathbf{M}$, and distributes secret shares to each party. Otherwise, the dealer re-generates a share matrix in the proper manner. Consequently, for usage of these schemes, it is necessary to verify that the share matrix $\mathbf{M}$ as intended in the *offline phase*. However, the time complexity of the offline verification step is exponential in $N$, making it impractical for large $N$. Suppose that a matrix $\mathbf{M}$ is constructed using $\{0,1\}$-LSSS for $t$-out-of-$N$ threshold access structure, and the secret $\mathsf{sk}$ is distributed to $N$ parties. Roughly speaking, to achieve the security of $\{0,1\}$-LSSS, any group of $t-1$ parties cannot recover the secret, whereas any group of $t$ or more parties can successfully recover $\mathsf{sk}$. On the other hand, the total number of valid parties is at least $\binom{N}{t}$, which is exponential in $N$. Thus, to confirm the validity of $\{0,1\}$-LSSS, it takes exponential time in $N$.

This verification step is included in our TreeSSS since our scheme also depends on the probabilistic construction Theorem 4.6, resulting in additional time costs.

To address this issue, we revisit the lemmas related to success probability (at least $1/2$). The success probability (at least $1/2$) of Theorem 4.6 stems from Lemma 4.7 and Lemma 4.8.

To mitigate the extra computational overhead, we make a minor adjustment to Lemma 4.7 and the iteration level $L$, it leads to increase success probability of a threshold structure. More precisely, we add an additional parameter $\kappa$. 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 $\log_s\left(\frac{N+\kappa}{N}\right)$.

**Proposition 4.9** *Given Shamir secret sharing $\mathsf{SS}(\ell, s)$ and the iteration number $L \geq \log_{c_s} N + \log_s(N+\kappa) + O(1)$ where $c_s = \frac{\ell}{2^{\ell-1}} \cdot \binom{\ell-1}{s-1}$, $L$-TreeSSS for $\frac{N+1}{2}$-out-of-$N$ threshold structure satisfies the correctness (Definition 4.3) and the privacy (Definition 4.4) with at least $1-\frac{1}{2^\kappa}$ success probability and the number of secret shares is $\ell^L = O(N^{\log_{c_s}\ell} \cdot (N+\kappa)^{\log_s \ell})$.*

We now prove the Proposition 4.9. The proof is almost the same as the proof of Lemma 4.7.

*Proof (of Proposition 4.9).* We change upper bounds in Lemma 4.7. Let $B_S$ be a set of secret shares $\{\mathsf{share}_j^{(L)}\}_{j\in\bigcup_{P_i\in S} T_i}$ for some subset $S \subset P$ and $L \geq log_{c_s} N + log_s(N+\kappa) + O(1)$. Then, we will prove the following inequalities:

$$\Pr[\perp \leftarrow \mathsf{TreeSSS.Combine}(B_S)] < \frac{1}{2^{N+\kappa+1}} \text{ for any } S \subset P \text{ of size } \tfrac{N+1}{2} \text{ or more,}$$

$$\Pr[\hat{\mathsf{sk}} \leftarrow \mathsf{TreeSSS.Combine}(B_{S'})] < \frac{1}{2^{N+\kappa+1}} \text{ for any } S' \subset P \text{ of size less than } \tfrac{N+1}{2}.$$

If we get above inequalities, then the following holds

$$\Pr[\mathsf{TreeSSS.Combine} \text{ satisfies the correctness}] \geq 1 - \frac{1}{2^\kappa},$$

which completes the proof.

To this end, we revisit a previous proof Lemma 4.7. In this case, we want to amplify the probability from $\frac{15}{16}$ to $\frac{1}{2^{N+\kappa+1}}$.

Therefore, lower bound of levels $j$ to exceed $1-\frac{1}{2^{N+\kappa+1}}$ as follows:

$$\begin{aligned} s^j + 3 &\geq N+\kappa+1 \\ j &\geq \log_s(N+\kappa-2) \approx \log_s(N+\kappa) + O(1). \end{aligned}$$

Therefore, if $L \geq \log_{c_s} N + \log_s(N+\kappa) + O(1)$, the inequalities we want to prove hold. □

Consequently, the number of shares can be changed into $O(N^{\log_{c_s}(2s-1)} \cdot (N+\kappa)^{\log_s(2s-1))})$, which is asymptotically equivalent to the number of shares in Theorem 4.6. We also note that in case of $\{0,1\}$-LSSS, $O(N^{5.3})$ is changed into $O(N^{3.3} \cdot (N+\kappa)^2)$. Table 2 gives results of this modifications.

For the purpose of clearer presentation, we opt not to use $\kappa$ throughout this paper, despite its probabilistic constraints.

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

| Secret Sharing Scheme | | Structure | $O(\log q)$ | # of keys |
|---|---|---|---|---|
| Previous | Shamir SS | $t$-out-of-$N$ | $O(N \log N)$ | $N$ |
| | $\{0,1\}$-LSSS | | $O(\log(N+\kappa))$ | $O(N^{3.3} \cdot (N+\kappa)^2)$ |
| TreeSSS | $\mathsf{SS}(3,2)$ | $t$-out-of-$N$ | $O(\log(N+\kappa))$ | $O(N^{2.72} \cdot (N+\kappa)^{1.58})$ |
| | $\mathsf{SS}(19,10)$ | | $O(\log(N+\kappa))$ | $O(N^{2.34} \cdot (N+\kappa)^{1.28})$ |
| | $\mathsf{SS}(99,50)$ | | $O(\log(N+\kappa))$ | $O(N^{2.22} \cdot (N+\kappa)^{1.17})$ |
| | $\mathsf{SS}(2s-1,s)$ | | $O(s \log s \cdot \log(N+\kappa))$ | $O(N^{2+\varepsilon} \cdot (N+\kappa)^{1+\varepsilon})$[11] |

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

*Remark 4.* We note that $\kappa$ is only appearing at $\log_s(\cdot)$. The proof of Proposition 4.9 consists of two parts. First, amplifying the probability from $\frac{1}{2} + \frac{1}{2N}$ to $\frac{15}{16}$. Next, amplifying the probability from $\frac{15}{16}$ to $1 - \frac{1}{2^{N+\kappa+1}}$. In first step, one needs $\log_{c_s}(N) + O(1)$ iterations of the Shamir secret sharing $\mathsf{SS}(2s-1,s)$. On the other hand, to obtain the probability $1 - \frac{1}{2^{N+\kappa+1}}$, one additionally needs $\log_s(N+\kappa) + O(1)$ iterations of $\mathsf{SS}(2s-1,s)$. In our computation, the total number of iterations is represented as follows.

$$\underbrace{\log_{c_s}(N) + O(1)}_{\text{First step of amplification}} + \underbrace{\log_s(N+\kappa) + O(1)}_{\text{Second step of amplification}}$$

This is bounded by $\log_{c_s}(N) + \log_s(N+\kappa) + O(1)$.

## 5 Theshold Fully Homomorphic Encryption

This section describes definitions of threshold fully homomorhpic encryption ($\mathsf{TFHE}$) and its construction based on TreeSSS.

### 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 ($\mathsf{TFHE}$))** *Let $\lambda$ be the security parameter and $d$ be a depth bound. Let $P = \{P_1, \ldots, P_N\}$ 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 $\lambda$, a depth bound $d$, and a threshold structure $\mathbb{A}$, and outputs $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$, where $\mathsf{pk}$ 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* $\mathsf{pk}$ *and a message* $\mu \in \{0,1\}$*, and outputs ciphertext* $\mathsf{ct}$*.*
- *The evaluation algorithm* $\mathsf{TFHE.Eval}(C, \mathsf{ct}_1, \ldots, \mathsf{ct}_l, \mathsf{pk})$ *takes as input a circuit of which depth is less than or equal* $d$*, a tuple of ciphertexts* $\mathsf{ct}_1, \ldots, \mathsf{ct}_l$ *and a public key* $\mathsf{pk}$*, 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* $\mathsf{pk}$*, a secret key share* $\mathsf{sk}_i$ *and the ciphertext* $\hat{\mathsf{ct}}$ *and outputs a partial decryption* $\mathsf{p}_i$ *related to the party* $P_i$*.*
- *The final decryption algorithm* $\mathsf{TFHE.FinDec}(\mathsf{pk}, B)$ *take as input a public key* $\mathsf{pk}$*, and a set* $B = \{\mathsf{p}_i\}_{i \in S}$ *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* $\mathsf{TFHE.Eval}(C, \mathsf{ct}_1, \ldots, \mathsf{ct}_l, \mathsf{pk})$ *satisfies*

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

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

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

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* $\lambda$ *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* $\lambda$ *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* $\mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_{N,t}) \to (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$*, and gives* $\mathsf{pk}$ *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs a set* $S \subset \{P_1, \ldots, P_N\}$ *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 $\lambda$, 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 $\lambda$ 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 $\mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_{N,t}) \to (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$, and gives $\mathsf{pk}$ to $\mathcal{A}$*
3. *$\mathcal{A}$ outputs a maximal invalid set $S^* \subset \{P_1, \ldots, P_N\}$ and messages $\mu_1, \ldots, \mu_k \in \{0,1\}$.*
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 $(S \subset \{P_1, \ldots, P_N\}, C)$ for circuits $C : \{0,1\}^k \to \{0,1\}$ of depth at most $d$. For each query, $\mathcal{C}$ computes $\hat{ct} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and provides $\{\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \hat{ct})\}_{i \in S}$ 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 $\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A}_{N,t}) \to (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N, \mathsf{st})$, and gives $\mathsf{pk}$ 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 $(S \subset \{P_1, \ldots, P_N\}, C)$, where $C : \{0,1\}^k \to \{0,1\}$ is a circuit of depth at most $d$. For each query, $\mathcal{C}$ runs the simulator*

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

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

### 5.2 TFHE using TreeSSS

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

- Let FHE be a special fully homomorphic encryption scheme (Definition 3.10) with noise bound $B$ and multiplicative constant $((2s-1)!)^L$ where $L \geq \log_{c_s} N + \log_s N + O(1)$ and $c_s$ is $\frac{2s-1}{4^{s-1}} \cdot \binom{2s-2}{s-1}$ for a positive integer $s \geq 2$.
- Let TreeSSS be a level-$L$ tree secret sharing scheme built from $s$-out-of-$2s-1$ Shamir 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 [40]. 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:

- $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_{N,t})$ :
  1. Sample $(\mathsf{fhepk}, \mathsf{fhesk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$.
  2. Compute $(\mathsf{share}_1^{(L)}, \ldots, \mathsf{share}_{(2s-1)^L}^{(L)}) \leftarrow \mathsf{TreeSSS.Share}(\mathsf{fhesk}, \mathbb{A}_{N,t})$.
  3. Distribute the secret shares to each party $P_i$ and define a index set of each party $T_i := \{j \mid P_i \text{ has } \mathsf{share}_j^{(L)}\}$.
  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 $\mathsf{ct}$.
- $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k, \mathsf{pk})$: Compute $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k, \mathsf{pk})$ and return $\hat{\mathsf{ct}}$.
- $\mathsf{p}_i \leftarrow \mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \hat{\mathsf{ct}})$:
  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 $\mathsf{p}_i = \{\hat{\mathbf{p}}_j^{(L)}\}_{j \in T_i}$ 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 (Compactness)** *Suppose FHE is a compact fully homomorphic encryption scheme. Then, the TFHE scheme in Construction 5.6 satisfies compactness.*

*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. □

**Theorem 5.8 (Correctness)** *Suppose FHE is a special fully homomorphic encryption scheme that satisfies correctness with noise bound $B$ and 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.*

*Proof (of Theorem 5.8).* By Construction 5.6, the following satisfies:

- Given the secret key of fully homomorphic encryption $\mathsf{fhesk}$, 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 $\mathsf{p}_i = \{\hat{\mathbf{p}}_j{}^{(L)}\}_{j \in T_i}$, 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 $j \in T_i = \{j \mid P_i \textit{ has } \mathsf{share}_j^{(L)}\}$ and any (valid) ciphertext $\mathsf{ct}$.

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 $\mathsf{sk}$ 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 $1 \le i \le L$, $T^{(i)} = \{k \mid \mathsf{share}_k^{(i)}$ can be recovered by $\{\mathsf{share}_j^{(L)}\}_{j \in T^{(L)}}\}$. Then, the correctness of $\mathsf{TreeSSS}$, $\mathsf{fhesk}$ can be expressed as follows:

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

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 secret sharing $\mathsf{SS}(2s-1, s)$.

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

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

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

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 3.10.

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 2.2, 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. □

**Theorem 5.9 (Security)** *Suppose FHE is a fully homomorphic encryption scheme that satisfies security and* TreeSSS *is a tree secret sharing scheme that satisfies the correctness. Then, TFHE scheme from Construction 5.6 satisfies semantic security.*

*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 $S \notin \mathbb{A}_{N,t}$. This means that the security of FHE implies the semantic security of TFHE. □

**Theorem 5.10 (Simulation security)** *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 $\mathcal{A}$'s 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 $C(\mu_1, \ldots, \mu_k)$ for each query $(S, C)$. The partial decryption algorithm takes in $(\mathsf{pk}, \hat{\mathsf{ct}}, \mathsf{sk}_i)$ and outputs $\mathsf{p}_i = \{\hat{\mathbf{p}}_j^{(L)}\}_{j\in T_i}$, 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

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

  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 $\mathsf{sk}_i$. This is an ideal experiment $Expt_{\mathcal{A},ideal}(1^\lambda, 1^d)$ of $\mathsf{TFHE}$ 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 $\mathsf{FHE}$ and definition of special $\mathsf{FHE}$, it holds that $\frac{q}{2} \cdot C(\mu_1, \ldots, \mu_k) = \mathsf{FHE.Dec}_0(\mathsf{sk}, \hat{\mathsf{ct}}) + \tilde{e}$ 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 $\hat{\mathbf{p}}_j^{(L)}$

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

$$= \mathsf{FHE.Dec}_0(\mathsf{share}_j, \hat{\mathsf{ct}}) + (c_j)^{-1} \cdot \tilde{e} + \mathbf{c} \cdot e_j.$$

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 2.2 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 implies 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.

**Theorem 6.1 ([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, one can generate an universal thresholdizer scheme such that compactness, evaluation correctness, verification correctness and security.*

## 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. Jean-Philippe Bossuat, Rosario Cammarota, Jung Hee Cheon, Ilaria Chillotti, Benjamin R Curtis, Wei Dai, Huijing Gong, Erin Hales, Duhyeong Kim, Bryan Kumara, et al. Security guidelines for implementing homomorphic encryption. *Cryptology ePrint Archive*, 2024.
16. Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from lwe with polynomial modulus. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023*, pages 371–404, Singapore, 2023. Springer Nature Singapore.
17. 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.
18. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.
19. Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key fhe with short ciphertexts. In *Annual Cryptology Conference*, pages 190–213. Springer, 2016.
20. 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.
21. 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.
22. 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.
23. 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.
24. 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.
25. 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.
26. 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.
27. Yvo Desmedt and Yair Frankel. Threshold cryptosystesns. *Advance in Cryptology*, pages 305–315, 1989.
28. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144, 2012.

29. 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.
30. 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.
31. 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.
32. Rosario Gennaro and Steven Goldfeder. One round threshold ecdsa with identifiable abort. *Cryptology ePrint Archive*, 2020.
33. Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security. In *International Conference on Applied Cryptography and Network Security*, pages 156–174. Springer, 2016.
34. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold dss signatures. *Information and Computation*, 164(1):54–84, 2001.
35. Oded Goldreich. *On (Valiant's) Polynomial-Size Monotone Formula for Majority*, pages 17–23. Springer International Publishing, Cham, 2020.
36. 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.
37. Arvind Gupta and Sanjeev Mahajan. Using amplification to compute majority with small majority gates. *Computational Complexity*, 6(1):46–63, 1996.
38. Kamil Doruk Gur, Jonathan Katz, and Tjerand Silde. Two-round threshold lattice signatures from threshold homomorphic encryption. Cryptology ePrint Archive, Paper 2023/1318, 2023. https://eprint.iacr.org/2023/1318.
39. 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.
40. Aayush Jain, Peter MR Rasmussen, and Amit Sahai. Threshold fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2017:257, 2017.
41. 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.
42. 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.
43. 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.
44. Yehuda Lindell. Fast secure two-party ecdsa signing. In *Annual International Cryptology Conference*, pages 613–644. Springer, 2017.
45. 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.
46. Philip MacKenzie and Michael K Reiter. Two-party generation of dsa signatures. *International Journal of Information Security*, 2(3):218–239, 2004.

47. 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.
48. Chris Peikert and Sina Shiehian. Multi-key fhe from lwe, revisited. In *Theory of Cryptography Conference*, pages 217–238. Springer, 2016.
49. 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.
50. Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
51. Victor Shoup. Practical threshold signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 207–220. Springer, 2000.
52. 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.
53. Leslie G. Valiant. Short monotone formulae for the majority function. *Journal of Algorithms*, 5(3):363–366, 1984.
54. 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.

## A About Approximation

We first introduce a useful inequalities to provide an approximation that we used. According to [22], $c_s$ is bounded by

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

Then, we have the following series of inequalities. From an upper bound of $c_s$, we get

$$\log_{c_s}(2s-1) + \log_s(2s-1) \leq \log_{2\sqrt{\frac{s-1/2}{\pi}}}(2s-1) + \log_s(2s)$$

Since $2s-1$ is represented by $\sqrt{\frac{s-1/2}{\pi}}^2 \cdot \frac{\pi}{2}$ and $\log_s(2s) = 1+\log_s 2$, the right-hand side is represented by

$$2 + \log_{2\sqrt{\frac{s-1/2}{\pi}}}(\pi/2) + (1 + \log_s 2).$$

Since we only consider $s \geq 2$, it holds that $\frac{(s-1/2)}{\pi} \geq s/4$, which implies

$$2\sqrt{\frac{s-1/2}{\pi}} \geq \sqrt{s}.$$

Thus, we have

$$2 + \log_{2\sqrt{\frac{s-1/2}{\pi}}}(\pi/2) + (1 + \log_s 2) \leq \log_{\sqrt{s}}(\pi/2) + 3 + \frac{1}{\log s}.$$

Last, using $\log \frac{\pi}{2} = 0.65149612947$, we have

$$\begin{aligned}\log_{\sqrt{s}}(\pi/2) + 3 + \frac{1}{\log s} &\leq \frac{1.30299}{\log s} + 3 + \frac{1}{\log s} \\ &\leq 3 + \frac{2.30299}{\log s}.\end{aligned}$$

Consequently, we have

$$\frac{1}{\sqrt{\pi}} \cdot \frac{2s-1}{\sqrt{s-1/2}} \leq 3 + \frac{2.30299}{\log s}$$

## B Observation of $\{0,1\}$-LSSS with [53] construction

$\{0,1\}$-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 [43] was proposed as an instantiation of $\{0,1\}$-LSSS. However, the polynomial-sized expression of threshold functions was proven by Valiant and Goldreich [35, 53]. Recently, [40] 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 $\varphi$ be a level-0 formula which takes $N$ bit-strings as input and returns one of the $i$-th input bits with some probability, where $i$ is randomly chosen, or returns 0. For each $i \geq 1$, the level-$(i+1)$ formula is defined as $\varphi = (\varphi_1 \wedge \varphi_2) \vee (\varphi_3 \wedge \varphi_4)$, with $\varphi_1, \varphi_2, \varphi_3, \varphi_4$ randomly selected from a family of level-$i$ formulas. Note that to maintain independence, the level-$i$ formulas will not be duplicated.

In classic works [35, 53], 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, [40] 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 $\ell$ 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. $\{0,1\}$-LSSS for threshold functions in [40] is constructed as follows:

1. Consider level-0 formulas $\varphi_i$, where $i \in [O(N^{5.3})]$.

2. Create a level-$(i+1)$ formula $\varphi$ 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, 40].

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 $\mathsf{count}$. 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 [40].

### B.1 Regarding $\{0,1\}$-LSSS as iterations of matrices

We reinterpret a secret sharing algorithm for threshold functions by utilizing the iterative steps of Boolean formula construction described in [53]. This allows us to construct a share matrix $\mathbf{M}$ through iterative matrix multiplications.

[53] 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 claim 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 Fig. 5. Since this binary tree is composed of AND and OR gates, we can directly apply the

folklore algorithm to the tree. As a result, 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 defined by

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

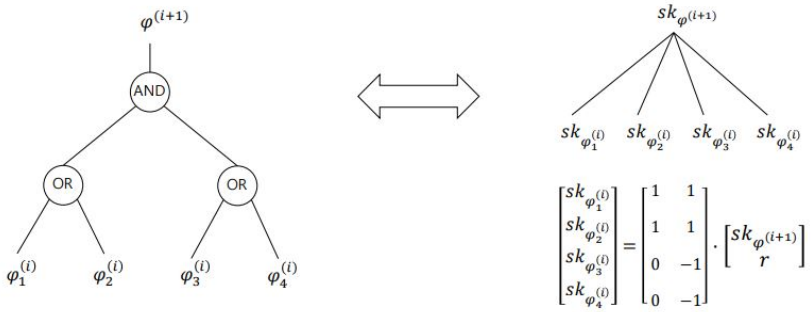



Fig. 5: Boolean formula corresponds to secret share

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. Consequently, there is a matrix $\mathbf{M}$ which corresponds to circuit representations of level-$t$ formula $\varphi^{(t)}$ from level-0 $\varphi^{(0)}$ formulas.

By the mathematical induction, we obtain a share matrix $\mathbf{M}$ of $\{0,1\}$-LSSS.[12] Furthermore, Share algorithm of $\{0,1\}$-LSSS is regarded by computing $\mathbf{M} \cdot \mathbf{v}$ for some $\mathbf{v}$.

[12] The proof is exactly the same as that of Proposition 4.5 except for using $\mathbf{D}$ rather than $\mathbf{V}_s$.