Threshold Fully Homomorphic Encryption

Aayush Jain* Peter M. R. Rasmussen* Amit Sahai*

March 20, 2017

Abstract

We formally define and give the first construction of (leveled) threshold fully homomorphic encryption for any access structure induced by a monotone boolean formula and in particular for the threshold access structure. Our construction is based on the learning with errors assumption and can be instantiated with any existing homomorphic encryption scheme that satisfies fairly general conditions, such as Gentry, Sahai, Waters (CRYPTO 2013) and Brakerski, Gentry, Vaikuntanathan (ITCS 2012).

From threshold homomorphic encryption, we construct function secret sharing and distributed pseudorandom functions for the aforementioned access structures. No such constructions were known prior to this work.

1 Introduction

The last few years have seen exciting developments that have changed the landscape of cryptography. Starting with the first breakthrough construction by [Gen09], Fully Homomorphic Encryption (FHE) has given rise to very surprising applications in the cryptographic world such as round efficient MPC [MW16, AJL+12, GLS15] and delegation for P [KRR14]. An FHE scheme allows arbitrary computations on encrypted data to be carried out without decryption. Not only has FHE helped realize many great primities, the techniques involved have themselves proven to be very useful. In particular, the techniques developed in a series of works on homomorphic encryption [Gen09, BV11, BGV12, GSW13, LTV12] helped to construct a multi-linear maps candidate [GGH13a], which allowed for the construction of the first candidate indistinguishability obfuscator [GGH+13b].

*UCLA and Center for Encrypted Functionalities. {aayushjain, rasmussen, sahai}@cs.ucla.edu. Research supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

In this work we study a natural question of constructing a "distributed" variant of FHE. More specifically, we consider a situation with multiple parties that each have a share of a secret key corresponding to an FHE public key. We wish to construct schemes that satisfy this with as general an access structure as possible.

We present a lighthearted example to illustrate this paradigm and the importance of flexible access structures. Say that the UN general assembly has collected data from all over the globe on occurrences of penguins. The data is stored as FHE encryptions in a database with each member state having a share of the secret key. In order to decrypt the result of a computation on the data, all member states must combine their key shares. Now, a team of researchers have analyzed the data in relation with the effort to save the penguins of the world. They need every country to supply their secret key share to decrypt and reveal the results. However, the country of Baroque vehemently denies the existence of penguins and refuses to provide its key towards the cause, thus blocking the endeavor. In such a situation the assembly must of course reconsider their data access policy. It is not fair nor practical that a single state or party can block a decryption and further, there is the issue of a key share being corrupted or lost.

Thus, it would be much better if more democratic and general access policies were possible. In the case of the UN general assembly, a majority access structure would be very practical for instance. More intricate access structures can be necessary as well. Suppose that a council on penguin welfare has been assembled. Then we might like an access structure that divides the power to decrypt between the general assembly and the council such that for instance the following groups can decrypt:

- Two members of the council and majority of the general assembly.
- A majority of the council.
- A two-thirds majority of the general assembly.

While the above example is rather innocent, one can imagine many real world situations in which intricate access structures such as those described would be necessary. In this work, we thus consider a very general question and try to achieve as broad a class of access structures as possible. Following the secret sharing literature, we ask the following of our scheme for some access structure $\mathbb{A}$.

1. A dealer generates an FHE public key pk and secret key shares $sk_1, \ldots, sk_n$ that are distributed among n parties.
2. Given a ciphertext ct for a message μ and a secret key share sk_i a party can compute a partial decryption p_i.
3. Given partial decryptions $\{p_i\}_{i \in S}$ for a set of parties S satisfying $\mathbb{A}$, the message μ can be recovered.

4. We require that any adversary that corrupts a subset of the parties $S \notin \mathbb{A}$ should not learn anything about the encrypted message even after seeing partial decryptions of other ciphertexts by the honest parties.[1]

We call a scheme satisfying these criteria a threshold homomorphic encryption (TFHE) scheme. Similar settings have been considered before, see for example [MW16, AJL+12, GLS15]. These works capture a somewhat different scenario. In [AJL+12, GLS15] there is a protocol for computing a joint public key and a distributed decryption. The work of [MW16] removed the bottleneck of having to run a protocol to generate a joint public key. However, both these works fail to capture our example as they have in common that all n parties must participate to decrypt the ciphertext.

In this paper, we consider a slightly different but equally natural question: can one instead induce a more general access structure $\mathbb{A}$ on the set of parties? Towards this goal, we achieve the following results.

Theorem 1. *Assuming LWE, there exists a threshold fully homomorphic encryption scheme for access structures induced by any monotone boolean formula.*

We combine this result with the work of [Val84], which constructs monotone boolean formulas for majority and t-threshold circuits, to get.

Corollary 1. *Assuming LWE, there exists a threshold fully homomorphic encryption scheme for the majority and t-out-of-n access structures for any t.*

Our techniques can also be applied elsewhere. For example, they can be used to construct threshold FHE for the access structure given by undirected s-t connectivity graphs which is compatible with the notion of evolving access structures [KNY16]. Moreover, we study some applications such as:

1.0.1 Function Secret Sharing [BGI15, BGI16, DHRW16].

The notion of function secret sharing was recently introduced in [BGI15]. In function secret sharing, a dealer secret shares a function f into parts $f_1, \ldots, f_n$. It is required that $\{f_i(x)\}_{i\in[n]}$ forms an additive secret sharing of $f(x)$. As pointed out in [BGI15], function secret sharing has a long list of applications such as homomorphic secret sharing, distributed point function, and multi-server PIR with secure keyword search. We generalize this notion to arbitrary access structures. From a TFHE scheme for a class of access structures $\mathbb{S}$, we give a construction of function secret sharing for $\mathbb{S}$.

1.0.2 Distributed PRF [BLMR13, Nie02, NPR99, NR04, MS95].

A distributed PRF is an algorithm that allows a PRF key k to be secret shared among a set of parties. Each party can output an additive share of the PRF computation for

[1] There are some technical nuances to this definition which we defer to a later section.

any input x. These shares can then be combined to learn $\mathsf{PRF}(k, x)$. Very importantly, evaluating the PRF can done without reconstructing the key in any single location. There has been a huge body of work in this area. Up until [BLMR13] all previous constructions either required multiple rounds of interaction or random oracle or interaction amongst key servers. The work of [BLMR13] gave a very clean and efficient construction from LWE using a key homomorphic PRF. Their construction has just one round and there is no interaction amongst parties. We give an alternate route to building such one round distributed PRFs: using threshold FHE. Using this approach, we are able to support more general access structures such as monotone boolean formulas.

We believe that our techniques may either directly apply or with some work lead to various other interesting notions for these access structures such as distributed point functions and homomorphic secret sharing.

2 Technical Overview

The goal of this paper is to construct a threshold homomorphic encryption scheme for all polynomial sized circuits and natural monotone access structures (such as t-out-of-n access structures and other access structures induced by monotone boolean formulas, undirected s-t graphs, and monotone span programs). Care has to be taken while defining such a primitive. Roughly speaking the requirement is twofold: First, we require semantic security to hold even when secret key shares are leaked for some unqualified set not satisfying the access structure. Second, there exists a simulator that statistically simulates the entire view of an adversary who possesses some unqualified set of secret shares in an adaptive game of requesting and receiving partial decryptions of ciphertexts.

2.0.1 Overall Approach.

Let FHE be a homomorphic encryption scheme with the following properties, which are satisfied by multiple FHE schemes such as [GSW13, BV11, BGV12].

1. For a prime p, the secret key sk is a vector $sk \in Z_q^n$.
2. The decryption function consists of two procedures $\mathsf{Decode} = (\mathsf{Decode}_0, \mathsf{Decode}_1)$.

   $\mathsf{Decode}_0(sk, ct)$**:** takes as input a secret key sk and a ciphertext ct and computes a function linear in sk to output $\mu \lceil q/2 \rceil + e$ where $e \in [-B, B]$ is an error term for some $B << q$ and $\mu \in \{0, 1\}$ is the message of the ciphertext ct.

   Decode_1**:** Properly rounds the output of Decode_0 to recover μ.

Since Decode_0 is linear in the key sk, the following idea is not too far fetched. Simply secret share sk with a linear secret sharing scheme Π for some access structure $\mathbb{A}$ on a set

of parties $P = \{P_1, \ldots, P_N\}$ into $sk_1, \ldots, sk_N$.[2] For any set $S \subseteq P$ with $S \in \mathbb{A}$, it is then the case that there are coefficients $\{c_i\}_{i \in S}$ such that

$$sk = \sum_{i \in S} c_i sk_i$$

and then by linearity of Decode_0, it follows that

$$\sum_{i \in S} c_i \mathsf{Decode}_0(sk_i, ct) = \mu \left\lceil \frac{q}{2} \right\rceil + e.$$

Thus, by secret sharing the secret key of the FHE scheme, the outputs of decoding with the secret key shares become a secret sharing of the decryption of the ciphertext. However, reconstructing the secret reveals the noise of the FHE scheme, which ultimately allows for reconstruction of the secret key sk after seeing just a few decryptions. To avoid this, it is necessary for the partial decryption $p_i' = \mathsf{Decode}_0(sk_i, ct)$ to have "smudging" noise added to it before being published as the share of party P_i. is the partial decryption

$$p_i = p_i' + e^{sm} \tag{1}$$

where e^{sm} is sampled from a distribution with variance much greater than e in order to statistically hide the noise e. This technique is called noise flooding and is used in several papers related to LWE. In the following we will describe how the above general approach leads to a rather intricate and surprising construction of TFHE for a large class of access structures. We start by presenting some more straightforward approaches that would intuitively work, but all run into obstacles.

2.0.2 Attempt 1: The Naive approach.

Given the above fairly intuitive outline of a construction for TFHE from an FHE scheme, one would think that constructing TFHE for a linear access structure should be trivial. However, this is far from the case. Let us consider the t-out-of-n access structure which is also the access structure for Shamir secret sharing [].

As an example, instantiating the above construction with Shamir secret sharing for the t-out-of-n access structure runs into the following problem. For $t < N$ the recovery coefficient c_i of the secret key shares can be very large, and in that case attempting to recover μ from the partial decryptions $\{p_i\}_{i \in [t]}$ with recovery coefficients $\{c_i\}_{i \in [t]}$ might satisfy that $sk = \sum_{i \in S} c_i sk_i$, but when attempting to recover μ, the smudging error overflows and ruins the correctness of the scheme as

$$\sum_{i \in [t]} c_i p_i = \sum_{i \in [t]} c_i p_i' + \sum_{i \in [t]} c_i e_i^{sm} \tag{2}$$

[2] See section 3.3 for an introduction to the terminology of secret sharing.

where $\sum_{i\in[t]} c_i e_i^{sm}$ can be arbitrarily large depending on the c_i. Actually, the scheme already works if $t = n$ as then each reconstruction coefficient is 1, but in the general case the approach fails as we observe that the reconstruction coefficients need to be small.

2.0.3 Attempt 2: Trying Small Coefficients.

To avoid the problem encountered above, one could try to use a linear secret sharing scheme with small reconstruction coefficients. Consider a Shamir style secret sharing scheme for t-out-of-n with reconstruction coefficients smaller than some constant $c = (N!)^2$ (which for instance is described in [BLMR13] in the context of constructing distributed PRFs from noisy key-homomorphic PRFs). Then generating partial decryptions as in (1) will preserve correctness of the scheme since now in equation (2) we have $\left|\sum_{i\in[t]} c_i e_i^{sm}\right| \leq tcB_{sm}$ where B_{sm} is the bound on the noise e^{sm} and the c_i are small.

However, now there is a problem with simulation. Let $\mathbb{A}$ be the access structure of our t-out-of-n scheme supporting small recovery coefficients and let $S \notin \mathbb{A}$ be a subset of parties. We will assume S to be of maximal size such that adding any new party to S would make it a valid share set. Now, given the secret key shares $\{sk_i\}_{i\in S}$ and the fact that the ciphertext ct decrypts to μ it should be possible to statistically simulate the partial decryptions of all the other parties. To do so, first generate $\{p'_i = \mathsf{Decode}_0(sk_i, ct)\}_{i\in S}$. Then for some party $P_k \notin S$, the only relation that is know regarding the value $p'_k = \mathsf{Decode}_0(sk_k, ct)$ is that for some valid subset $S' \subseteq S \cup \{P_k\}$ (which must contain P_k) and small recovery coefficients $\{c_i\}_{i\in S'}$,

$$\sum_{i\in S'} p'_i = \mu \left\lceil \frac{q}{2} \right\rfloor + e.$$

In order to simulate the partial decryption $p_k = p'_k + e^{sm}$, one would need to derive p'_k from this expression by computing

$$p'_k = c_k^{-1} \left(\mu \left\lceil \frac{q}{2} \right\rfloor + e - \sum_{i\in S'} p'_i \right).$$

However, knowing only an invalid set of secret key shares, the simulator doesn't know the noise e. Thus, the closest approximation of the value p'_k must be

$$\tilde{p}_k = c_k^{-1} \left(\mu \left\lceil \frac{q}{2} \right\rfloor - \sum_{i\in S'} p'_i \right),$$

which differs from the real value p'_k by $\left|c_k^{-1} e\right|$, which can be large since while c_k is small, no such guarantees exists for c_k^{-1}. This shows that for simulation security to hold, we would want the reconstruction coefficients to both be small and have small inverses. We can ensure this if the reconstruction is additive.

2.0.4 Attempt 3: Splitting Shares.

A natural approach to solving the problem of the above attempts is to have each party output multiple partial shares such that there always is a share set with recovery coefficients in $\{0, 1\}$. We will demonstrate this technique and an attack on it in the following

Consider again as above the Shamir secret sharing scheme for t-out-of-n secret sharing the secret key sk of the FHE scheme into shares $sk_1, \ldots, sk_N$ such that party P_i receives the share sk_i. Let q be the prime modulus of the FHE scheme. To partially decrypt, first compute $p'_i = \mathsf{FHE.Decode}_0(sk_i, ct)$. Instead of outputting $p'_i + e_i$ where $e_i \xleftarrow{\$} [-B_{sm}, B_{sm}]$ as the partial decryption, we instead sample $e_i^j \xleftarrow{\$} [-B_{sm}, B_{sm}]$ for every $0 \leq j \leq \log(q)$ and output

$$\{p_i^j = 2^j p'_i + e_i^j\}_{j=0}^{\lfloor \log(q) \rfloor}.$$

Now, reconstruction can be done using recovery coefficients $c_i \in \{0, 1\}$. To see this, say that in the original Shamir secret sharing scheme, the coefficients of recovery were $c_i, i \in [t]$. Then finding $c_{i,j} \in \{0, 1\}$ such that $c_i = \sum_{j=0}^{\lfloor \log(q) \rfloor} 2^j c_{i,j}$, we can recover the plaintext of the ciphertext ct as

$$\sum_{i=1}^{t} \sum_{j=0}^{\lfloor \log(q) \rfloor} c_{i,j} p_i^j = \mu \left\lceil \frac{q}{2} \right\rfloor + e + \sum_{i=1}^{t} \sum_{j=0}^{\lceil \log(q) \rceil} e_i^j.$$

Having such a scheme, however, does not help us for two reasons. First, recovering and simulating the partial decryptions given an invalid share set runs into the same problems as before. It is not clear how one would simulate partial decryptions since given μ and partial keys $\{sk_i\}_{i \in [t] \setminus \{k\}}$ since the equation would be

$$c_k p'_k = \sum_{j=0}^{\lfloor \log(q) \rfloor} 2^j c_{k,j} p'_k = \mu \left\lceil \frac{q}{2} \right\rfloor - \sum_{i \in [t] \setminus \{k\}} \sum_{j=0}^{\lfloor \log(q) \rfloor} 2^j c_{i,j} p'_i.$$

Second, splitting the shares in this manner not only makes simulation difficult, but also leads to direct attacks. In fact, it can be shown that letting each e_i^j be drawn from the interval $[-B_{sm}, B_{sm}]$ where $B_{sm} < q/2$ allows for complete recovery of p'_i given $\{p_i^j = 2^j p'_i + e_{i,j}\}_{j=0}^{\lfloor \log(q) \rfloor}$.

2.0.5 Solution: TFHE for a Special Access Structure.

Let us step back for a moment and observe the restrictions we have derived, what it would take for a secret sharing scheme to be compatible with our construction. The properties are as follows.

1. The secret sharing is a linear secret sharing scheme over some prime field Z_q which is compatible with the FHE scheme.

2. A recovery coefficients c_i of the scheme can be chosen from $\{0,1\}$ such that both c_i and c_i^{-1} are small.

3. The share of each party and the partial decryptions are both single elements of $\mathbb{Z}_q$.

We emphasize that this property fails for most commonly used secret sharing schemes such as Shamir secret sharing (as shown above). In our paper, we define $\{0,1\}$-LSSS to be the class of access structures that allow for a secret sharing scheme satisfying these properties (see Section 3.3.1 for details). The most important attribute of $\{0,1\}$-LSSS is that it lets the simulation security proof that previously failed go through.b

More specifically, applying the construction of our overall approach to a secret sharing scheme Π for an access structure $\mathbb{A} \in \{0,1\}$-LSSS will yield a TFHE scheme. In this TFHE scheme correctness holds because of the small recovery coefficients as demonstrated above. Further, simulation security will now go through as follows. Say that a simulator holds secret keys shares $\{sk_i\}_{i\in S}$ for an invalid share set $S \notin \mathbb{A}$ of parties, which is maximal in the sense that adding any new party to S would yield a valid share set. Given a ciphertext ct, knowledge of the decryption μ of ct, and again computing $\{p'_i = \mathsf{Decode}_0(sk_i, ct)\}_{i\in S}$, the simulator can now statistically simulate the partial decryption of any other party P_k. Since the set $S \cup \{P_k\}$ is valid and the recovery coefficients of the scheme Π belong to $\{0,1\}$ there exists a set $S' \subseteq S \cup \{P_k\}$ (which will contain P_k) such that $\sum_{i\in S'} sk_i = sk$ and thus,

$$p'_k = \mu \left\lceil \frac{q}{2} \right\rceil + e - \sum_{i \in S' \setminus \{k\}} p'_i.$$

Not knowing e, the simulator can only compute $\tilde{p}_i = \mu \left\lceil \frac{q}{2} \right\rceil - \sum_{i\in S'\setminus\{k\}} p'_i$, which will only differ from p'_k by $|e|$, which is small. Thus, adding the "smudging" noise, $p'_k + e^{sm}$ and $\tilde{p}_k + e^{sm}$ are statistically close.

Thus, we manage to construct TFHE for the $\{0,1\}$-LSSS class of access structures.

2.0.6 Extending Solution: Monotone Formulas and Majority.

While having TFHE for some access structure is good, the class $\{0,1\}$-LSSS seems a strange one. Two obvious access structures which are contained in $\{0,1\}$-LSSS are undirected s-t-connectivity and n-out-of-n, but other than those, it is not obvious which other useful access structures are.

However, it turns out that the class is in fact fairly large. We show that in fact the access structures defined by the monotone boolean formulas with input fan-out 1 (*special* monotone boolean circuits) belong to $\{0,1\}$-LSSS through a folklore algorithm. Said algorithm converts a special monotone boolean formula into an LSSS share matrix as follows. Consider a special monotone formula C as a binary tree T whose nodes are either AND or OR gates. Now, label the nodes of T with vectors $v \in \mathbb{Z}_q^*$ recursively:

1. Label the root node r by $v_r = (1)$ (the vector consisting of a single index of value 1).

2. For every node s labeled with the vector v_s label its children r and l (padding all constructed labels with 0s after each operation to make all labels of equal length):
   (a) If s is an OR gate, label r and l by $v_r = v_l = v_s$.
   (b) If s is an AND gate, label r by $v_r = (v_s \,||\, 1)$ and l by $v_l = (0, \ldots, 0, -1)$ (of the same length as v_r).

The matrix with rows corresponding to the labels of the leafs of T is now an LSSS share matrix for the access structure induced by C.

We prove that this construction yields a secret sharing scheme where the reconstruction coefficients can always be chosen in $\{0, 1\}$ by considering minimal valid share sets, i.e. valid share sets of parties such that removing any party renders the set invalid, and by induction on the depth of the C.

Now, it would be natural to extend a construction of TFHE for special monotone boolean formulas to one for regular monotone boolean formulas. First, note that the above algorithm while yielding an LSSS share matrix would not yield a share matrix for $\{0, 1\}$-LSSS if given a regular monotone boolean formula, since each party would need to hold multiple shares if the inputs had input fan-out greater than one, contradicting the definition of $\{0, 1\}$-LSSS. Further, it does not seem possible to merge multiple shares into a single field element. Thus, we employ a different strategy. Consider a monotone boolean $C\colon \{0,1\}^N \to \{0,1\}$ with multiple input fan-out. Assuming without loss of generality that every input of C has fan-out ℓ, generate a new special monotone boolean formula $C'\colon \{0,1\}^{\ell N} \to \{0,1\}$. The formula C' is derived from C by letting every fan-out of an input gate of C be its own input in C'. Then C' is a special monotone boolean formula, and we can describe the access structure it induces with a $\{0, 1\}$-LSSS scheme.

While the behavior of C' is possibly very different from that of C, the circuit C can clearly still be evaluated from C' by setting all the input wires of C' that previously were a single input of C to the same value. Thus, if in the access structure induced by C we have parties $\{Q_1, \ldots, Q_{\ell N}\}$ corresponding to the inputs of C' then letting them collude, such that for instance party P_1 of a new access structure corresponds to the parties $\{Q_1, \ldots, Q_\ell\}$, we can create the access structure induced by C. It follows that if we can allow collusion between parties in $\{0, 1\}$-LSSS and still have security of the TFHE scheme then we can realize TFHE for the access structure induced by C.

To prove that collusion between parties of a TFHE scheme for $\{0, 1\}$-LSSS does not violate security, we generalize our simulation. Say that parties $P_1, \ldots, P_N$ each hold shares $\{sk_1^i, \ldots, sk_\ell^i\}$ instead of just a single share for shares $\{sk_j^i\}_{j\in[\ell], i\in[N]}$ generated from a $\{0, 1\}$-LSSS scheme for parties $Q_1, \ldots, Q_{\ell N}$. Now, given shares for a maximal invalid share set of parties $\{P_i\}_{i\in S}$ we can we can no longer guarantee that the parties of the underlying $\{0, 1\}$-LSSS scheme, $\bigcup_{i\in S}\{sk_j^i\}_{j\in[\ell]}$, form a maximal invalid share set. To solve this problem, the simulator simply simulates a maximal invalid share set of secret key shares containing the shares of $\bigcup_{i\in S}\{sk_j^i\}_{j\in[\ell]}$. These shares are possible to simulate through the properties

of secret sharing in general. Now, since as shown above, the simulator can statistically simulate each individual partial decryption of the underlying $\{0,1\}$-LSSS scheme, we can show that in fact the simulator can statistically simulate the partial decryption $\{p_j^i\}_{j\in[\ell]}$ of party P_i.

In this manner, we achieve TFHE for the access structure of monotone boolean formulas. Now, a result by [Val84] shows a construction for construction of the majority access structure from monotone boolean formulas. Using this we now easily achieve TFHE even for the t-out-of-n access structure.

2.0.7 Applications.

We consider the following two applications of a threshold FHE:

- **Function Secret Sharing:** Given any levelled TFHE for access structure class $\mathbb{S}$ we construct function secret sharing scheme for all circuits and the same access structure class $\mathbb{S}$. Some connection of threshold FHE to function secret sharing was already observed in [BGI15, DHRW16, AJN+16]. The construction is very simple: Each function share consists of a TFHE encryption ct of f along with a key sk_i (and a PRF key K_i to compute random computations deterministically). Evaluation on any input x consists of just computing $\mathsf{TFHE.PartDec}(sk_i, ct_x)$ where ct_x is the ciphertext encrypting $f(x)$ computed using ct.

- **Distributed PRF:** Given any levelled TFHE for access structure class $\mathbb{S}$ and a PRF g we construct distributed PRF for the same access structure class. The construction is basically the same as the function secret sharing scheme defined above except that now we encrypt the PRF key. However, the proof is more subtle and deviates from the proof for the scheme above. The details can be found in Section 8.2. We note that in a similar manner we can also construct a distributed point function.

3 Preliminaries

Notation 1. *For any set X, we denote by $\mathcal{P}(X)$ the powerset of X.*

Throughout this work, we will denote the security parameter by λ. For any $Y, Z \in \{0,1\}^n$ we say that $Y \subseteq Z$ if it happens that for each $i \in [n]$ such that $Y_i = 1$, $Z_i = 1$

3.1 Homomorphic Encryption

The construction in this paper is based on a traditional, generic homomorphic encryption scheme satisfying a few additional properties. As observed by [MW16], these properties can be found for the schemes [GSW13]. Other schemes such as [BV11, BGV12] can also be adapted to satisfy these properties.

Definition 1 (FHE). *A fully homomorphic encryption* (FHE) *scheme is a tuple of PPT algorithms*

$$\mathsf{FHE} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Eval}, \mathsf{Decode})$$

satisfying the following specifications:

- $(pk, sk) \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$*: The setup algorithm takes as input the security parameter and a depth bound* d *and outputs a key pair* (pk, sk)*.*
- $ct \leftarrow \mathsf{Encrypt}(pk, \mu)$*: The encrypt algorithm takes as input a bit* $\mu \in \{0,1\}$ *and the public key* pk*. It outputs* ct*.*
- $\hat{ct} \leftarrow \mathsf{Eval}(C, ct_1, .., ct_k)$*: The deterministic evaluate algorithm takes as input a tuple of ciphertexts* $ct_1, .., ct_k$ *and a circuit* $C \in \mathcal{C}_\lambda$ *of depth less than or equal* d *and outputs an evaluated ciphertext* $\hat{ct}$*.*
- $\mu^* \leftarrow \mathsf{Decode}(sk, ct)$*: The deterministic decryption algorithm takes as input a ciphertext and a secret key and outputs* $\mu^* \in \{0, 1, \perp\}$*.*

We ask that the algorithms satisfy the following:

Correctness of Evaluation *Let* $(pk, sk) = \mathsf{Setup}(1^\lambda, 1^d)$*,* $ct_i = \mathsf{Encrypt}(pk, \mu_i)$ *for* $1 \leq i \leq k$ *and* $\mu_i \in \{0,1\}$*,* $ct_{k+1} = \mathsf{Eval}(C, ct_1, \ldots, ct_k)$ *for* $C \in \mathcal{C}_\lambda$*. With all but negligible probability in* λ *over the coins of* Setup*,* Encrypt*, and,*

$$\mathsf{Decode}(sk, ct_i) = \begin{cases} C(\mu_1, \ldots, \mu_k), & i = k+1 \\ \mu_i & \textit{Otherwise.} \end{cases}$$

Compactness of Ciphertexts *There exists a polynomial,* poly*, such that* $|ct| \leq \mathsf{poly}(\lambda, d)$ *for any ciphertext* ct *generated from the algorithms of* FHE *(possibly after evaluation).*

Semantic Security of Encryption *Any PPT adversary* $\mathcal{A}$ *has only negligible advantage as a function of* λ *over the coins of all the algorithms in the following game:*

1. *The adversary is given as input the security parameter* 1^λ *and a circuit depth* 1^d*.*
2. *Run* $\mathsf{Setup}(1^\lambda, 1^d) \to (pk, sk)$*. The adversary is given* pk*.*
3. *The adversary receives* $\mathsf{Encrypt}(pk, b) \to ct$ *for a random* $b \in \{0,1\}$*.*
4. *The adversary outputs* b' *and wins if* $b = b'$*.*

Having defined generic FHE, we ask further in our case that the FHE scheme satisfies the following.

1. On input the 1^λ and 1^d the setup algorithm outputs (pk, sk) where the public key contains a prime q and the secret key sk is a vector $sk \in Z_q^m$ for some $m = \mathsf{poly}(\lambda, d)$.

2. The decryption algorithm consists of two functions $\mathsf{Decode} = (\mathsf{Decode}_0, \mathsf{Decode}_1)$.

   $p \leftarrow \mathsf{Decode}_0(sk, ct)$ takes as input the ciphertext encrypting $\mu \in \{0,1\}$ and secret key vector and outputs $p \in Z_q$, where $p = \mu \lceil q/2 \rceil + e$ for $e \in [-B, B]$ and $B = B(\lambda, d, q)$.

   $\mu \leftarrow \mathsf{Decode}_1(p)$ takes as input p and computes the appropriate rounding function to recover $\mu \in \{0,1\}$.

3. Decode_0 is a linear operation over $\mathbb{Z}_q$ in the secret key.

3.2 Statistical Distance

This section establishes results related to statistical closeness of distributions. Looking ahead these will be used in the security proof of our TFHE construction. Some of the proofs and related material has been placed in the appendix.

Definition 2 (Statistical Distance)**.** *Let E be a finite set, Ω a probability space, and $X, Y \colon \Omega \to E$ random variables. We define the* statistical distance *between X and Y to be the function Δ defined by*

$$\Delta(X, Y) = \frac{1}{2} \sum_{e \in E} \left| \Pr_X(X = e) - \Pr_Y(Y = e) \right|.$$

Lemma 1. *Let E be a finite set and Ω a probability space. The function Δ is a metric on the set of random variables $Z \colon \Omega \to E$.*

Proof. Consider the variables $X, Y \colon \Omega \to E$ to be finite dimensional vectors indexed by the elements of E and with each index e having the value $\Pr_X(X = e)$ and $\Pr_Y(Y = e)$, respectively. Then Δ is simply the norm induced by scaling the ℓ_1 norm on an $|E|$-dimensional space by $\frac{1}{2}$. The conclusion follows. □

Definition 3 (Conditioned Statistical Distance)**.** *Let E be a finite set and Ω a probability space. Let random variables $A_1, \ldots, A_n \colon \Omega \to E$ and $X, Y \colon \Omega \to E$ be given. We define the statistical distance of X and Y conditioned on the variables $A_1, \ldots, A_n$ to be*

$$\Delta_{\{A_i\}_{i \in [n]}}(X, Y) = \max_{(a_1, \ldots, a_n) \in E^n} \frac{1}{2} \sum_{e \in E} \Big| \Pr_X \left(X = e \mid (A_i)_{i \in [n]} = (a_i)_{i \in [n]} \right) - \Pr_Y \left(Y = e \mid (A_i)_{i \in [n]} = (a_i)_{i \in [n]} \right) \Big|.$$

Before moving on, we need the following lemma.

Lemma 2. *Let E be a finite set, Ω a probability space, and $\{X_i^b\}_{i\in[n],b\in\{0,1\}} : \Omega \to E$ random variables such that for every $i \in [n]$, every $(e_1,\dots,e_i) \in E^i$, and every $(b_1,\dots,b_{i-1}),(b'_1,\dots,b'_{i-1}) \in \{0,1\}^{i-1}$,*

$$\Pr\left(X_i^0 = e_i \mid X_1^{b_1} = e_1,\dots,X_{i-1}^{b_{i-1}} = e_{i-1}\right) = \Pr\left(X_i^0 = e_i \mid X_1^{b'_1} = e_1,\dots,X_{i-1}^{b'_{i-1}} = e_{i-1}\right),$$
$$\Pr\left(X_i^1 = e_i \mid X_1^{b_1} = e_1,\dots,X_{i-1}^{b_{i-1}} = e_{i-1}\right) = \Pr\left(X_i^1 = e_i \mid X_1^{b'_1} = e_1,\dots,X_{i-1}^{b'_{i-1}} = e_{i-1}\right).$$

Then the joint distributions $(X_1^0,\dots,X_n^0)$ and $(X_1^1,\dots,X_n^1)$ satisfy

$$\Delta((X_1^0,\dots,X_n^0),(X_1^1,\dots,X_n^1)) \le \sum_{i=1}^{n} \Delta_{\{X_j^0\}_{j<i}}(X_i^0,X_i^1).$$

Proof. Define the vector $Z_i = (X_1^0,\dots,X_i^0,X_{i+1}^1,\dots,X_n^1)$ for each $0 \le i \le n$. We will now show that for $1 \le i \le n$, $\Delta(Z_{i-1},Z_n) \le \Delta_{\{X_j^0,X_j^1\}_{j<i}}(X_i^0,X_i^1)(X_i^0,X_i^1)$. From this the result will follow by the triangle inequality since Δ is a metric on the set of random variables $\Omega \to E^n$ by Lemma 1 and $Z_0 = (X_1^1,\dots,X_n^1)$ and $Z_n = (X_1^0,\dots,X_n^0)$. Now,

$$\Delta(Z_{i-1},Z_n) = \frac{1}{2}\sum_{(e_1,\dots,e_n)\in E^n} \left|\Pr\left(Z_{i-1} = (e_1,\dots,e_n)\right) - \Pr\left(Z_i = (e_1,\dots,e_n)\right)\right|,$$

and using $e_{[a:b]} = (e_a,e_{a+1},\dots,e_b)$ we can define

$$\begin{aligned}
A(e_{[1,i-1]}) &= \Pr\left((X_1^0,\dots,X_{i-1}^0) = e_{[1,i-1]}\right)\\
B_0(e_{[1,i]}) &= \Pr\left(X_i^0 = e_i \mid (X_1^0,\dots,X_{i-1}^0) = e_{[1,i-1]}\right)\\
B_1(e_{[1,i]}) &= \Pr\left(X_i^1 = e_i \mid (X_1^0,\dots,X_{i-1}^0) = e_{[1,i-1]}\right)\\
C(e_{[1,n]}) &= \Pr\left((X_{i+1}^1,\dots,X_n^1) = e_{[i+1,n]} \mid (X_1^0,\dots,X_{i-1}^0,X_i^1) = e_{[1,i]}\right)\\
&= \Pr\left((X_{i+1}^1,\dots,X_n^1) = e_{[i+1,n]} \mid (X_1^0,\dots,X_{i-1}^0,X_i^0) = e_{[1,i]}\right)
\end{aligned}$$

where the last equality follows from the assumption of the Lemma. With this notation, we can write

$$\begin{aligned}
\Pr\left(Z_{i-1} = e_{[1,n]}\right) &= A(e_{[1,i-1]})B_1(e_{[1,i]})C(e_{[1,n]})\\
\Pr\left(Z_i = e_{[1,n]}\right) &= A(e_{[1,i-1]})B_0(e_{[1,i]})C(e_{[1,n]})
\end{aligned}$$

and get

$$\begin{aligned}
\Delta(Z_{i-1}, Z_n) &= \frac{1}{2} \sum_{\vec{e} \in E^n} \left| A(e_{[1,i-1]}) B_1(e_{[1,i]}) C(e_{[1,n]}) - A(e_{[1,i-1]}) B_0(e_{[1,i]}) C(e_{[1,n]}) \right| \\
&= \frac{1}{2} \sum_{\vec{e}_{[1,i-1]} \in E^{i-1}} A(e_{[1,i-1]}) \sum_{\vec{e}_i \in E} \left| B_1(e_{[1,i]}) - B_0(e_{[1,i]}) \right| \sum_{e_{[i+1,n]} \in E^{n-i}} C(e_{[1,n]}) \\
&= \sum_{\vec{e}_{[1,i-1]} \in E^{i-1}} A(e_{[1,i-1]}) \frac{1}{2} \sum_{\vec{e}_i \in E} \left| B_1(e_{[1,i]}) - B_0(e_{[1,i]}) \right| \\
&\leq \Delta_{\{X_j^0, X_j^1\}_{j<i}}(X_i^0, X_i^1) \sum_{\vec{e}_{[1,i-1]} \in E^{i-1}} A(e_{[1,i-1]}) \\
&= \Delta_{\{X_j^0, X_j^1\}_{j<i}}(X_i^0, X_i^1).
\end{aligned}$$

since $C(e_{[1,n]})$ is dependent on $e_{[i+1,n]}$ and conditioned on $e_{[1,i]}$, and similarly $B_0(e_{[1,i]})$ and $B_1(e_{[1,i]})$ are dependent on e_i and conditioned on $e_{[1,i-1]}$. □

The above lemma is simply a more general form of the following result, which describes a rather general occurrence in cryptography, which is usually dealt with by a series of hybrids in the security proof, but which we in this presentation instead will state and prove formally in order to make the proof of security more concise.

Proposition 1. *Let E be a finite set, Ω a probability space, and let $\{X_s^b\}_{s \in S, b \in \{0,1\}}$ be a family of random variables $X_s^b \colon \Omega \to E$ indexed by an element $s \in S$ and a state $b \in \{0,1\}$. Further, assume that for every $s \in S$ we have $\Delta(X_s^0, X_s^1) \leq \delta$. Now, for a stateful PPT algorithm $\mathcal{A}$, define the following experiment:*

$\mathsf{Expt}_{\mathcal{A},b,m}$:

- *The algorithm $\mathcal{A}$ issues m queries. Each query is an element $s_i \in S$ and after each query, $\mathcal{A}$ receives in return $x_i \leftarrow X_{s_i}^b$ sampled independently of the other samples.*
- *The output of the experiment is $(s_1, x_1), \ldots, (s_m, x_m)$.*

Then $\Delta(\mathsf{Expt}_{\mathcal{A},0,m}, \mathsf{Expt}_{\mathcal{A},1,m}) \leq m\delta$.

Proof. Define a new random variable $Y_i^b \colon \Omega \to S \times E$ which is sampled as follows. On the ith query, the adversary outputs a seed based on the previous queries, $s_i \leftarrow \mathcal{A}((s_1, x_1), \ldots, (s_{i-1}, x_{i-1}))$. Then $x_i \leftarrow X_{s_i}^b$ is sampled independently of the other samplings. It is clear that the following distributions are equal

$$\mathsf{Expt}_{\mathcal{A},b,m} = (Y_1^b, \ldots, Y_m^b).$$

We make two observations. First, the algorithm $\mathcal{A}$ does not see which distribution each x_i is sampled from and the value $x_i \leftarrow X_{s_i}^b$ is always sampled independently of the other samplings, thus, changing b between 0 and 1 from query to query is irrelevant to the outcome of s_i as long as the sampled values are the same. I.e. we have for every $i \in [n]$, every $((s_1, x_1), \ldots, (s_i, x_i)) \in (S \times E)^i$, and every $(b_1, \ldots, b_{i-1}), (b'_1, \ldots, b'_{i-1}) \in \{0,1\}^{i-1}$ that

$$\Pr\left(Y_i^0 = (s_i, x_i) \mid \{Y_j^{b_1}\}_{j<i} = \{(s_j, x_j)\}_{j<i}\right) = \Pr\left(Y_i^0 = (s_i, x_i) \mid \{Y_j^{b'_1}\}_{j<i} = \{(s_j, x_j)\}_{j<i}\right),$$
$$\Pr\left(Y_i^1 = (s_i, x_i) \mid \{Y_j^{b_1}\}_{j<i} = \{(s_j, x_j)\}_{j<i}\right) = \Pr\left(Y_i^1 = (s_i, x_i) \mid \{Y_j^{b'_1}\}_{j<i} = \{(s_j, x_j)\}_{j<i}\right).$$

Second, $\Delta_{\{Y_j^0\}_{j<i}}(Y_i^0, Y_i^1) \leq \delta$ since the distribution of the first component is the same for the two distributions as the algorithm $\mathcal{A}$ only gets to know the outcome of the previous queries and the second parameter, sampled from $X_{s_i}^b$ independently of the other samplings, has statistical distance less than δ.

The result now follows from Lemma 2. □

Another useful lemma is the following, which demonstrates a technique to "smudge" or hide the presence of error (e_1 in the lemma) by adding a much larger error. While no values are specifically given in the statement of the lemma, B_1 is meant to be negligible compared to B_2 such that the statistical distance between the two distributions is negligible.

Lemma 3 (Smudging Lemma [MW16])**.** *Let $B_1, B_2 \in \mathbb{N}$. For any $e_1 \in [-B_1, B_1]$ let E_1 and E_2 be independent random variables uniformly distributed on $[-B_2, B_2]$ and define the two stochastic variables $X_1 = E_1 + e_1$ and $X_2 = E_2$. Then $\Delta(E_1, E_2) < B_1/B_2$.*

Proof. Noting that X_1 and X_2 can only take values on $[e_1 - B_2, e_1 + B_2]$ and $[-B_2, B_2]$, respectively, we get the statistical distance

$$\Delta(X_1, X_2) = \frac{1}{2} \sum_{k \in \mathbb{Z}} \left| \Pr_{X_1}(X_1 = k) - \Pr_{X_2}(X_2 = k) \right| = |e_1/B_2| < B_1/B_2. \tag{3}$$

□

Notation 2 (Smudging noise)**.** *In the following we will denote by χ_B^{sm} the uniform distribution on the interval $[-B, B]$.*

3.3 Secret Sharing

Throughout this paper we will use secret sharing terminology and techniques. This section provides an introduction to the basic terms, notation, and concepts that will be needed later.

3.3.1 Secret Sharing Basics.

We assume that the reader if familiar with the notion of a information theoretic secret sharing scheme and in particular the shamir secret sharing scheme. We now describe concepts about access structures and specific secret sharing schemes that we consider in this paper. We adapt some definitions from [LW11].

Definition 4 (Monotone Access Structure)**.** *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. A collection $\mathbb{A} \subseteq \mathcal{P}(P)$ is* monotone *if whenever we have sets B, C satisfying $B \in \mathbb{A}$ and $B \subseteq C \subseteq P$ then $C \in \mathbb{A}$. A* monotone access *structure on P is a monotone collection $\mathbb{A} \subseteq \mathcal{P}(P) \setminus \emptyset$. The sets in $\mathbb{A}$ are called the* valid *sets and the sets in $\mathcal{P}(P) \setminus \mathbb{A}$ are called the* invalid *sets.*

For ease of notation, we will generally identify a party with its index. Further, since this presentation will only consider monotone access structures, the terms monotone access structure and simply access structure will be used interchangeably throughout the text.

Notation 3. *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and let S be a subset of P. We denote by $\mathbf{X}_S$ the vector $\mathbf{X}_S = (x_1, \ldots, x_N)$ where $x_i = 1$ if $P_i \in S$ and $x_i = 0$ otherwise.*

Definition 5 (Efficient Access Structure)**.** *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and $\mathbb{A} \subseteq \mathcal{P}(P)$ a monotone access structure on P. We say that $\mathbb{A}$ is* efficient *if there exists a polynomial size circuit $f_{\mathbb{A}} \colon \{0,1\}^N \to \{0,1\}$ such that for all $S \subseteq P$, $f_{\mathbb{A}}(\mathbf{X}_S) = 1$ if and only if $S \in \mathbb{A}$.*

Definition 6 (Class of Monotone Access Structures)**.** *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. A* class of monotone access structures *is a collection $\mathbb{S} = \{\mathbb{A}_1, \ldots, \mathbb{A}_t\} \subseteq \mathcal{P}(\mathcal{P}(P))$ of monotone access structures on P.*

Being interested in secret sharing, we will only consider efficient access structures in this work.

One of the most canonical classes of access structures is the t-out-of-n class.

Definition 7 (t-out-of-n secret sharing)**.** *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. An access structure $\mathbb{A}$ is a t-out-of-n access structure if for every $S \subseteq P$, $S \in \mathbb{A}$ if and only if $|S| \geq t$.*

A more general form of secret sharing is linear secret sharing.

Definition 8 (Linear Secret Sharing Scheme (LSSS))**.** *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. The class of access structures LSSS (or LSSS_N to emphasize the number of parties) consists of all access structure $\mathbb{A}$ such that there exists a secret sharing scheme Π satisfying:*

1. *For a prime p, the share of each party P_i is a vector $\vec{w}_i \in \mathbb{Z}_p^{n_i}$ for some $n_i \in \mathbb{N}$.*

2. *There exists a matrix* $M \in \mathbb{Z}_p^{\ell \times n}, \ell = \sum_{i=1}^{N} n_i$ *called the* share matrix *for* Π *with size polynomial in the number of parties and such that for a secret* s*, the shares are generated as follows. Values* $r_2, \ldots, r_n \in \mathbb{Z}_p$ *are chosen at random and the vector* $\vec{v} = M \cdot (s, r_2, \ldots, r_n)^T$ *is generated. Now, denote by* $T_i \subseteq [\ell], 1 \leq i \leq N$ *a partition of* $[\ell]$ *such that* T_i *has size* $|T_i| = n_i$ *and is associated with party* P_i*. The share of* P_i *is the vector* $\vec{w}_i = (v_i)_{i \in T_i}$*.*

3. *For any set* $S \subseteq P$*,* $S \in \mathbb{A}$ *if and only if*

$$(1, 0, \ldots, 0) \in \mathsf{span}(\{M[i]\}_{i \in \bigcup_{j \in S} T_j})$$

over $\mathbb{Z}_p$ *where* $M[i]$ *denotes the* i*th row of* M*.*

We denote by LSSS_N *the class of linear secret sharing schemes on* N *parties.*

Note that keeping with the notation of the LSSS definition above, any set of parties $S \subseteq P$ such that $S \in \mathbb{A}$ can recover the secret by finding coefficients $\{c_i\}_{i \in \bigcup_{j \in S} T_j}$ satisfying

$$\sum_{i \in \bigcup_{j \in S} T_j} c_i M[i] = (1, 0, \ldots, 0).$$

Given such coefficients, the secret can be recovered simply by computing

$$s = \sum_{i \in \bigcup_{j \in S} T_j} c_i v_i.$$

Since such coefficients can be found in time polynomial in the size of M using linear algebra, LSSS is a class of efficient access structures [Bei96]. Further, LSSS has the property that it information theoretically hides the value s, i.e. For any secrets $s_0$and s_1, it holds that the distributions of shares $\{\vec{w}_i\}_{i \in S}$ for a set $S \notin \mathbb{A}$, are identical.

In our application of linear secret sharing, we will always be sharing a vector over $\mathbb{Z}_p$, $\vec{s} \in \mathbb{Z}_p^n$ instead of just a single element of $\mathbb{Z}_p$. Simply linearly secret sharing each entry of the vector $\vec{s}$ using fresh randomness for each entry yields shares $\vec{s}_1, \ldots, \vec{s}_\ell \in \mathbb{Z}_p^n$. It is easy to see that the secret $\vec{s} \in \mathbb{Z}_q^n$ can now be reconstructed as a linear combination of the shares $\vec{s}_i$ using the same coefficients as for a single field element. Further, information theoretical hiding is maintained.

3.3.2 $\{0,1\}$-LSSS and $\{0,1\}$-LSSSD.

For the purposes of this paper, we will need a more restricted class of access structures. The access structures of this class can be realized as LSSS schemes such that each party only has one share and such that it always is possible to only use recovery coefficients $c_i \in \{0, 1\}$.

Definition 9 ($\{0,1\}$-Linear Single Share Scheme ($\{0,1\}$-LSSS)). *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. The set $\{0,1\}$-$\mathsf{LSSS}_N \subseteq \mathsf{LSSS}_N$ is the collection of access structures $\mathbb{A} \in \mathsf{LSSS}_N$ such that there exists an efficient linear secret sharing scheme Π for $\mathbb{A}$ satisfying the following:*

1. *For a prime p, the share of each party P_i consists of a single element $w_i \in \mathbb{Z}_p$.*
2. *Let s be a secret and let $w_i \in \mathbb{Z}_p$ be the share of party P_i for each i. For every valid set $S \in \mathbb{A}$, there exist a subset $S' \subseteq S$ such that $s = \sum_{i \in S'} w_i$.*

In our application, we will need $\{0,1\}$-LSSS schemes that work over a certain prime q corresponding to the modulus of an FHE scheme. The constructions of later sections will be designed in a way that allows for the access structure to work over any modulus, but for now we will denote by $\{0,1\}$-LSSS^q the set of access structures contained in $\{0,1\}$-LSSS that can be realized as secret sharing schemes by a share matrix over $\mathbb{Z}_q$.

That every access structure $\mathbb{A} \in \{0,1\}$-LSSS is efficient follows directly from the efficiency of the LSSS class. However, it is not obvious that the set S' of the above definition can be found efficiently given any $S \subseteq P$. To see that this is indeed the case, we first establish a lemma.

Definition 10 (Maximal Invalid Share Set). *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and $\mathbb{A}$ be a monotone access structure on P. A set $S \subseteq P$ is a* maximal invalid share set *if $S \notin \mathbb{A}$ but for every $p \in P \setminus S$, $S \cup \{p\} \in \mathbb{A}$.*

Definition 11 (Minimal Valid Share Set). *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and $\mathbb{A}$ be a monotone access structure on P. A set $S \subseteq P$ is a* minimal valid share set *if $S \in \mathbb{A}$ and for every $S' \subsetneq S$, $S' \notin \mathbb{A}$.*

Although the following lemma is trivial to show it will turn out to be a useful observation both for the efficiency of reconstruction of $\{0,1\}$-LSSS and for our construction.

Lemma 4. *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties, $\mathbb{A} \in \{0,1\}$-LSSS, and Π be a linear secret sharing scheme as specified in the definition of $\{0,1\}$-LSSS. Let s be a secret, let $w_i \in \mathbb{Z}_p$ be the share of party P_i for each i, and let $S \subseteq P$ be a minimal valid share set of $\mathbb{A}$. Then $s = \sum_{i \in S} w_i$.*

Finally, the following lemma shows that given a linear secret sharing scheme Pi for $\mathbb{A} \in \{0,1\}$-LSSS, we can find recovery coefficients efficiently. However, it is worth noting that this does not mean that deciding whether an access structure belongs to $\{0,1\}$-LSSS is feasible. In our applications we will instead specifically construct secret sharing schemes that belong to $\{0,1\}$-LSSS.

Lemma 5. *Finding recovery coefficients $c_i \in \{0,1\}$ in a linear secret sharing scheme Π for an access structure $\mathbb{A} \in \{0,1\}$-LSSS can be done efficiently.*

Proof. Let $P = \{P_1, \ldots, P_N\}$ be a set of parties, let $\mathbb{A} \in \{0,1\}$-LSSS be an access structure on P, and let M be a share matrix for $\mathbb{A}$ of polynomial size as specified in Definition 9. By Lemma 4, $\mathbb{A}$ is efficient if for every $S \subseteq P$ such that $S \in \mathbb{A}$ we can find a minimal valid share set $S' \subseteq S$ in polynomial time in the size of the share matrix M and the number of parties N. This can be done as follows. Enumerate the elements of S as $S = \{P_{a_1}, P_{a_2}, \ldots, P_{a_k}\}$ and set $S_0 = S$. We will without loss of generality assume that row i is associated with P_i and recursively define

$$S_i = \begin{cases} S_i, & \text{if } (1,0,\ldots,0) \notin \mathsf{span}(\{M[j]\}_{j \in S_i \setminus \{P_{a_i}\}}), \\ S_i \setminus \{P_{a_i}\}, & \text{if } (1,0,\ldots,0) \in \mathsf{span}(\{M[j]\}_{j \in S_i \setminus \{P_{a_i}\}}), \end{cases}$$

for $1 \leq i \leq k$. It is easy to see that S_k is a minimal valid share set contained in S and that the above procedure runs in polynomial time in the size of the share matrix. □

In applications, we will need the following access structure, which removes the constraint on the number of shares per party, but retains the overall property of the shares.

Definition 12 (Derived $\{0,1\}$-LSSS ($\{0,1\}$-LSSSD))**.** *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. We denote by $\{0,1\}$-LSSSD_N the class of access structures $\mathbb{A} \in \mathsf{LSSS}_N$ such that there exists an $\ell \in \mathbb{N}$ polynomial in N and an access structure $\mathbb{B} \in \{0,1\}$-$\mathsf{LSSS}_{\ell n}$ for parties $P' = \{P'_1, \ldots, P'_{N\ell}\}$ such that we can associate the party $P_i \in P$ with the parties $P'_{\ell(i-1)}, P'_{\ell(i-1)+1}, \ldots, P'_{\ell i} \in P'$ as follows. For every $S \subseteq [N]$, $S \in \mathbb{A}$ if and only if the set S' of parties of P' associated with a party in S, $S' \in \mathbb{B}$. More precisely, for every $S \subseteq [N]$,*

$$\bigcup_{i \in S} \{P_i\} \in \mathbb{A} \text{ if and only if } \bigcup_{i \in S} \{P'_{\ell(i-1)}, P'_{\ell(i-1)+1}, \ldots, P'_{\ell i}\} \in \mathbb{B}. \tag{4}$$

In other words, a $\{0,1\}$-LSSSD scheme is a secret sharing scheme where the shares satisfy a $\{0,1\}$-LSSS scheme, but each party receives multiple shares.

4 Definition of Threshold Fully Homomorphic Encryption

In this section we present the definition of threshold fully homomorphic encryption for any class of access structures.

Definition 13 (TFHE)**.** *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and let $\mathbb{S}$ be a class of efficient access structures on P. A threshold fully homomorphic encryption scheme for $\mathbb{S}$ is tuple of PPT algorithms*

$$\mathsf{TFHE} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$$

satisfying the following specifications:

$(pk, sk_1, \ldots, sk_N) \leftarrow \mathsf{Setup}(1^\lambda, 1^d, \mathbb{A})$**:** *Takes as input a security parameter λ, a circuit depth d, and an access structure $\mathbb{A}$ on P. Outputs a public key pk, and secret key shares $sk_1, \ldots, sk_N$. We shall assume that all the following algorithms also takes as input the public key.*

$c \leftarrow \mathsf{Encrypt}(pk, \mu)$**:** *Takes as input a public key pk and a single bit plaintext $\mu \in \{0,1\}$. Outputs a ciphertext ct.*

$\widehat{ct} \leftarrow \mathsf{Eval}(C, ct_1, \ldots, ct_k)$**:** *Takes as input a boolean circuit $C\colon \{0,1\}^k \to \{0,1\} \in \mathcal{C}_\lambda$ of depth $\leq d$ and ciphertexts $ct_1, \ldots, ct_k$ encrypted under the same public key. Outputs an evaluation ciphertext $\widehat{ct}$.*

$p_i \leftarrow \mathsf{PartDec}(ct, sk_i)$**:** *Takes as input a ciphertext ct and a secret key share sk_i. Outputs a partial decryption p_i related to the party P_i.*

$\widehat{\mu} \leftarrow \mathsf{FinDec}(B)$**:** *Takes as input a set $B = \{p_i\}_{i \in S}$ for some $S \subseteq \{P_1, \ldots, P_N\}$ where we recall that we identify a party P_i with its index i. Deterministically outputs a plaintext $\widehat{\mu} \in \{0, 1, \perp\}$.*

We ask of TFHE *that for parameters $(pk, sk_1, \ldots, sk_N) \leftarrow \mathsf{Setup}(1^\lambda, 1^d, \mathbb{A})$, any plaintexts $\mu_1, \ldots, \mu_k \in \{0,1\}$, any subset $S \subseteq \{P_1, \ldots, P_N\}$, and any boolean circuit $C\colon \{0,1\}^k \to \{0,1\} \in \mathcal{C}_\lambda$ of depth $\leq d$ the following is satisfied.*

Correctness of Encryption. *Let $ct = \mathsf{Encrypt}(pk, \mu_1)$ and $B = \{\mathsf{PartDec}(ct, sk_i)\}_{i \in S}$. With all but negligible probability in λ over the coins of* Setup*,* Encrypt*, and* PartDec*,*

$$\mathsf{FinDec}(B) = \begin{cases} \mu_1, & S \in \mathbb{A} \\ \perp & S \notin \mathbb{A}. \end{cases}$$

Correctness of Evaluation. *Let $ct_i = \mathsf{Encrypt}(pk, \mu_i)$ for $1 \leq i \leq k$, $\hat{ct} = \mathsf{Eval}(C, ct_1, \ldots, ct_k)$, and $B = \{\mathsf{PartDec}(\hat{ct}, sk_i)\}_{i \in S}$. With all but negligible probability in λ over the coins of* Setup*,* Encrypt*, and* PartDec*,*

$$\mathsf{FinDec}(B) = \begin{cases} C(\mu_1, \ldots, \mu_k), & S \in \mathbb{A} \\ \perp & S \notin \mathbb{A}. \end{cases}$$

Compactness of Ciphertexts. *There exists a polynomial,* poly*, such that $|ct| \leq \mathsf{poly}(\lambda, d, N)$ for any ciphertext ct generated from the algorithms of* TFHE*.*

Semantic Security of Encryption. *Any PPT adversary $\mathcal{A}$ has only negligible advantage as a function of λ over the coins of all the algorithms in the following game:*

1. *On input the security parameter 1^λ and a circuit depth 1^d, the adversary $\mathcal{A}$ outputs $\mathbb{A} \in \mathbb{S}$.*

2. *Run* $\mathsf{Setup}(1^\lambda, 1^d, \mathbb{A}) \to (pk, sk_1, .., sk_N)$. *The adversary is given* pk.
3. *The adversary outputs a set* $S \subseteq \{P_1, \ldots, P_N\}$ *such that* $S \notin \mathbb{A}$.
4. *The adversary receives* $\{sk_i\}_{i \in S}$ *along with* $\mathsf{Encrypt}(pk, b) \to ct$ *for a random* $b \in \{0, 1\}$.
5. *The adversary outputs* b' *and wins if* $b = b'$.

Simulation Security. *There exists a stateful PPT algorithm* Sim *such that for any PPT adversary* $\mathcal{A}$*, we have that the experiments* $\mathsf{Expt}_{\mathcal{A},Real}(1^\lambda, 1^d)$ *and* $\mathsf{Expt}_{\mathcal{A},\mathsf{Sim}}(1^\lambda, 1^d)$ *as defined below are statistically close as a function of* λ *over the coins of* Setup, Encrypt, Eval, PartDec, Sim, *and* $\mathcal{A}$*. The experiments are defined as follows:*

$\mathsf{Expt}_{\mathcal{A},Real}(1^\lambda, 1^d)$**:**

1. *On input the security parameter* 1^λ *and a circuit depth* 1^d*, the adversary* $\mathcal{A}$ *outputs* $\mathbb{A} \in \mathbb{S}$.
2. *Run* $\mathsf{Setup}(1^\lambda, 1^d, C, N) \to (pk, sk_1, .., sk_N)$. *The adversary is given* pk.
3. *The adversary outputs a set* $S \subseteq \{P_1, \ldots, P_N\}$ *such that* $S \notin \mathbb{A}$ *together with plaintexts* $\mu_1, \ldots, \mu_k \in \{0, 1\}$. *The adversary is handed over* $\{sk_i\}_{i \in S}$
4. *For each* μ_i*, the adversary is given* $\mathsf{Encrypt}(pk, \mu_i) \to ct_i$.
5. *The adversary issues polynomially many queries of the form* $(S_i \subseteq \{P_1, \ldots, P_N\}, C_i)$ *for circuits* $C_i \colon \{0, 1\}^k \to \{0, 1\} \in \mathcal{C}_\lambda$. *After each query the adversary receives for each* $l \in S_i$ *the value*

$$\mathsf{PartDec}(\mathsf{Eval}(C_i, ct_1, \ldots, ct_k), sk_l) \to p_l.$$

6. *In the end,* $\mathcal{A}$ *outputs* out. *The output of the experiment is* out.

$\mathsf{Expt}_{\mathcal{A},\mathsf{Sim}}(1^\lambda, 1^d)$**:**

1. *On input the security parameter* 1^λ *and a circuit depth* 1^d*, the adversary* $\mathcal{A}$ *outputs* $\mathbb{A} \in \mathbb{S}$.
2. *Run* $\mathsf{Setup}(1^\lambda, 1^d, C, N) \to (pk, sk'_1, .., sk'_N)$. *The adversary is given* pk.
3. *The adversary outputs a set* $S \subseteq \{P_1, \ldots, P_N\}$ *such that* $S \notin \mathbb{A}$ *together with plaintexts* $\mu_1, \ldots, \mu_k \in \{0, 1\}$. *The Simulator Sim takes as input* $pk, \mathbb{A}, S$ *and outputs* $\{sk_i\}_{i \in S}$ *and the state* state. *The adversary is handed over* $\{sk_i\}_{i \in S}$.
4. *For each* μ_i*, the adversary is given* $\mathsf{Encrypt}(pk, \mu_i) \to ct_i$.
5. *The adversary issues polynomially many queries of the form* $(S_i \subseteq \{P_1, \ldots, P_N\}, C_i)$ *for circuits* $C_i \colon \{0, 1\}^k \to \{0, 1\} \in \mathcal{C}_\lambda$. *After each query, the simulator* Sim *computes*

$$\mathsf{Sim}(C, \{ct_l\}_{l=1}^k, C_i(\mu_1, .., \mu_k), S_i, \mathsf{state}) \to \{p_l\}_{l \in S_i}$$

and sends $\{p_l\}_{l \in S_i}$ *to* $\mathcal{A}$.

6. In the end, $\mathcal{A}$ outputs out*. The output of the experiment is* out*.*

Together the semantic security and simulation security definitions above imply the following natural indistinguishability based definition.

Definition 14 (IND-secure TFHE)**.** *Let* $\mathsf{TFHE} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$ *be a threshold fully homomorphic encryption scheme for* $\mathbb{S}$*, satisfying correctness and compactness requirement as described above. We say that* TFHE *is IND-secure if the advantage of any PPT adversary in the following security game is negligible in the security parameter.*

1. *On input the security parameter* 1^λ *and a circuit depth* 1^d *the adversary* $\mathcal{A}$ *outputs an access structure* $\mathbb{A} \in \mathbb{S}$*.*
2. *Run* $\mathsf{Setup}(1^\lambda, 1^d, \mathbb{A}) \to (pk, sk_1, .., sk_N)$*. The adversary is given* pk*.*
3. *The adversary outputs a set* $S \subseteq \{P_1, \ldots, P_N\}$ *such that* $S \notin \mathbb{A}$ *and* S *is a maximal invalid set.*
4. *The adversary receives* $\{sk_i\}_{i \in S}$*.*
5. $\mathcal{A}$ *now outputs two message vectors* $(\mu_1^0, .., \mu_k^0)$ *and* $(\mu_1^1, .., \mu_k^1)$*. The adversary is now given* $\mathsf{Encrypt}(pk, \mu_i^b) \to ct_i$ *for every* $i \in [k]$ *and a random* $b \in \{0, 1\}$*.*
6. *The adversary issues polynomially many queries of the form of circuits* $C_i \colon \{0,1\}^k \to \{0,1\} \in \mathcal{C}_\lambda$ *such that* $C_i(\mu_1^0, .., \mu_k^0) = C_i(\mu_1^1, .., \mu_k^1)$*. After each query the adversary receives the value*
$$\mathsf{PartDec}(\mathsf{Eval}(C_i, ct_1, \ldots, ct_k), sk_l) \to p_l$$
for each $l \in [N]$*.*
7. *The adversary outputs* b' *and wins if* $b = b'$*.*

Remark: We note that semantic security along with simulation security implies IND based security almost directly. This is because any scheme that satisfies semantic and simulation security also satisfies IND security. This can be shown by invoking the TFHE simulator in the IND game. The simulator uses $C_i(\mu_1^0, .., \mu_k^0) = C_i(\mu_1^1, .., \mu_k^1)$ to perform simulations. Then, invoke semantic security to encrypt $\mu_1^0, .., \mu_k^0$. At this point the game is independent of b.

5 TFHE for $\{0,1\}$-LSSSD

In this section we present a construction of threshold homomorphic encryption for the $\{0,1\}$-LSSSD class of access structures. We proceed by first establishing a Proposition.

Proposition 2. *Any* TFHE *for* $\{0,1\}$-LSSS *allows for* TFHE *for* $\{0,1\}$-LSSSD*.*

Proof. Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and $\mathbb{A} \in \{0,1\}$-LSSSD. Then by definition there exists ℓ and an access structure $\mathbb{B} \in \{0,1\}$-LSSS on parties $P' = \{P'_1, \ldots, P'_{\ell n}\}$ such that the party $P_i \in P$ is associated with parties $\{P'_{\ell(i-1)+1}, \ldots, P'_\ell\} \subseteq P'$ as in Definition 12. Then a TFHE scheme, TFHE_B, for $\mathbb{B}$ will extend to a TFHE scheme, TFHE_A, for $\mathbb{A}$ as follows. Each party $P_i \in P$ in TFHE_A acts as all the parties $P'_{\ell(i-1)+1}, \ldots, P'_\ell$ in TFHE_B simultaneously, receiving ℓ secret keys and outputting ℓ partial decryptions when queried.

All the security properties of TFHE_B will carry over to the derived TFHE_A as follows. First, correctness of encryption and evaluation and compactness of ciphertexts follow immediately. Second, semantic security follows since the only difference between the semantic security games of TFHE_A and TFHE_B is that in step 3 of the semantic security game of TFHE_B the adversary can request any subset $S \subseteq P'$ with $S \notin \mathbb{B}$, while in the semantic security game for TFHE_A, he can only ask for subsets $S' \subseteq P$ which are equivalent to the subsets

$$\overline{S}' = \bigcup_{i \in S'} \{P'_{\ell(i-1)+1}, \ldots, P'_{\ell i}\} \subseteq P'$$

with $\overline{S}' \notin \mathbb{B}$. Finally, simulation security follows in exactly the same was as semantic security, since any subset of secret key shares the adversary can ask for in the simulation security game of TFHE_A, the adversary can also request in the simulation security game of TFHE_B. □

Thus, for our purposes it suffices to construct TFHE for $\{0,1\}$-LSSS. The next sections will go over such a construction and its analysis.

5.1 Construction

We describe TFHE for the class of access structures $\{0,1\}$-LSSS.

Construction 1 ($\{\mathbf{0},\mathbf{1}\}$-LSSS TFHE)**.** *Let $P = \{P_1, \ldots, P_N\}$ be parties and* FHE *be a scheme satisfying all the properties of section 3.1. Then define the following algorithms*

$$\mathsf{TFHE} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$$

as follows:

$(pk, sk_1, \ldots, sk_N) \leftarrow \mathsf{Setup}(1^\lambda, 1^d, \mathbb{A})$**:** *Run* $\mathsf{FHE.Setup}(1^\lambda, 1^d) \to (pk, sk)$*, let the noise χ of the* FHE *scheme be B-bounded, and let $\mathbb{Z}_q$ be the field of our* FHE *scheme. Now, secret share the vector $sk \in \mathbb{Z}_q^n$ into secret shares $sk_1, \ldots, sk_N$ according to the scheme Π*[3]*. Output the public key pk and secret keys $(sk_1, \ldots, sk_N)$.*

[3]Note that here we assume that in fact $\mathbb{A} \in \{0,1\}$-LSSS^q. While we do not generally know how to ensure this, it will not be an issue for our later applications to monotone boolean formulas. See Section 6 and in particular Section 6.3 for details.

$ct \leftarrow \mathsf{Encrypt}(pk, \mu)$**:** *Encrypt the bit* $\mu \in \{0,1\}$ *using* pk *according to the FHE scheme* $\mathsf{FHE.Encrypt}(pk, \mu) \to ct$*. Output the ciphertext* ct*.*

$\widehat{ct} \leftarrow \mathsf{Eval}(C, ct_1, \ldots, ct_k)$**:** *Evaluate the circuit* C *on ciphertexts* $ct_1, \ldots, ct_k$ *by running* $\mathsf{FHE.Eval}(C, ct_1, \ldots, ct_k)$ $\widehat{ct}$*. Output the evaluated ciphertext* $\widehat{ct}$*.*

$p_i \leftarrow \mathsf{PartDec}(ct, sk_i)$**:** *Decrypt the ciphertext* ct *according to the* FHE *scheme under secret key* sk_i *to acquire* $\mathsf{FHE.Decode}_0(ct, sk_i) \to p_i'$ *and sample* $e^{sm} \xleftarrow{\$} \chi^{sm}_{B_{sm}}$*. Output* $p_i = p_i' + e^{sm}$*.*

$\mu \leftarrow \mathsf{FinDec}(\{p_i\}_{i \in S})$**:** *If* $S \notin \mathbb{A}$ *output* $\perp$*. Otherwise compute a minimal valid set for* $\mathbb{A}$*,* $S' \subseteq S$ *and compute* $\mathsf{FHE.Decode}_1(\sum_{i \in S'} p_i) \to \mu$*. Output the plaintext* μ*.*

5.2 Analysis

The following sections constitute the analysis of our $\{0,1\}$-LSSS scheme. Throughout, we let λ be the security parameter, d a circuit depth, and $\mathbb{A}$ a $\{0,1\}$-$LSSS_N$ scheme on the set of parties $P = \{P_1, \ldots, P_N\}$. We run the setup procedure $\mathsf{Setup}(1^\lambda, 1^d, \mathbb{A})$ to produce public key pk and secret key shares $sk_1, \ldots, sk_N$ of the secret key sk of the FHE scheme.

Correctness of Encryption Let a set S be given, let $ct = \mathsf{Encrypt}(pk, \mu)$ for some bit $\mu \in \{0,1\}$, and let $B = \{p_i = \mathsf{PartDec}(ct, sk_i)\}_{i \in S}$ where we denote by e_i^{sm} the value drawn from χ^{sm} while computing $\mathsf{PartDec}(ct, sk_i)$. If $S \notin \mathbb{A}$ then $\mathsf{FinDec}(B) = \perp$ directly by the definition of FinDec. On the other hand, if $S \in \mathbb{A}$, FinDec computes the minimal valid set $S' \subseteq S$ for $\mathbb{A}$, which by Lemma 4 satisfies

$$\sum_{i \in S'} sk_i = sk.$$

Thus, by the linearity of $\mathsf{FHE.Decode}_0$ we have

$$\begin{aligned}
\mathsf{FinDec}(B) &= \mathsf{FHE.Decode}_1\left(\sum_{i \in S'} p_i\right) \\
&= \mathsf{FHE.Decode}_1\left(\sum_{i \in S'} \mathsf{FHE.Decode}_0(ct, sk_i) + e_i^{sm}\right) \\
&= \mathsf{FHE.Decode}_1\left(\mathsf{FHE.Decode}_0(ct, \sum_{i \in S'} sk_i) + \sum_{i \in S'} e_i^{sm}\right) \\
&= \mathsf{FHE.Decode}_1\left(\mu \left\lceil \frac{q}{2} \right\rceil + e + \sum_{i \in S'} e_i^{sm}\right)
\end{aligned}$$

where e is the error incurred by the $\mathsf{FHE.Decode}_0$ algorithm. As long as

$$\widehat{E} = \left| e + \sum_{j \in J} e_{smudge,j} \right| < q/4$$

we have $\mathsf{FHE.Decode}_1\left(\mu \left\lceil \frac{q}{2} \right\rceil + e + \sum_{i \in S'} e_i^{sm}\right) = \mu$. Assuming $B + NB_{sm} < q/4$, $E < q/4$ with all but negligible probability in the security parameter over the coins of the algorithms and the conclusion follows.

Correctness of Evaluation Follows from the correctness of evaluation of the FHE scheme and the correctness of encryption shown above.

Compactness of Ciphertexts Follows from the compactness of ciphertexts of the FHE scheme.

5.2.1 Semantic Security.

In the semantic security game, the adversary is given secret key shares $\{sk_i\}_{i \in S}$ for a set $S \notin \mathbb{A}$. By the properties of $\{0,1\}$-LSSS, the value sk is information theoretically hidden given $\{sk_i\}_{i \in S}$, so any advantage the adversary would have in the semantic security game the adversary would have in the semantic security game of our FHE scheme.

5.2.2 Simulation Security.

We define our simulator Sim as follows with the two subroutines Setup and Query. Intuitively the simulator gets a set $S \notin \mathbb{A}$ and computes a maximal set $S' \notin \mathbb{A}$ containing S. It secret shares 0 according to $\mathbb{A}$ to simulate partial decryption keys for this set S'. Next, these keys are used to simulate any secret key queries. This can be done due to information theoretic security of the secret sharing scheme. The partial decryption queries are simulated by also using the ciphertext and the decrypted value in addition to the secret keys by the equation (5).

$(\{sk'_i\}_{i \in S}, \mathsf{state}) \leftarrow \mathsf{Sim.Setup}(pk, S, \mathbb{A})$ Given the public key pk, an invalid share set S, and an access structure $\mathbb{A}$, constructs a maximal invalid share set S' such that $S \subseteq S' \notin \mathbb{A}$. Create secret key shares $\{sk'_i\}_{i \in S'}$ of the zero-vector $\vec{0}$. Output $(\{sk'_i\}_{i \in S}, \mathsf{state} = (\{sk'_i\}_{i \in S'}, S'))$.

$\{\widehat{p}_i\}_{i \in T} \leftarrow \mathsf{Sim.Query}(C, \{ct_i\}_{i=1}^k, \mu, T, \mathsf{state}_0)$ Given the circuit C, ciphertexts $\{ct_i\}_{i=1}^k$, the evaluation μ, requested share set T, and initial state $\mathsf{state}_0 = (\{sk'_i\}_{i \in S'}, S')$, first compute $\tilde{p}_i = \mathsf{FHE.Decode}_0(\mathsf{Eval}(C, ct_1, \ldots, ct_k), sk'_i)$ for every $i \in S'$.

Second, for every $i \in T \setminus S'$ let $T'_i \subseteq S' \cup \{i\}$ be a minimal valid set (which will contain i) and compute

$$\tilde{p}_i = \mu \left\lceil \frac{q}{2} \right\rceil - \sum_{j \in T'_i \setminus \{i\}} \tilde{p}_j. \tag{5}$$

Finally, for every $i \in T$ sample $e_i^{sm} \xleftarrow{\$} \chi_{B_{sm}}^{sm}$ and compute $\widehat{p}_i = \tilde{p}_i + e_i^{sm}$. Output $\{\widehat{p}_i\}_{i \in T}$.

Theorem 2. *With the above simulator* Sim*, the outputs of the two experiments* $\mathsf{Expt}_{\mathcal{A},Real}(1^\lambda, 1^d)$ *and* $\mathsf{Expt}_{\mathcal{A},\mathsf{Sim}}(1^\lambda, 1^d)$ *are statistically close.*

Proof. We go through a series of hybrid experiments starting with hybrid_0, which is the real experiment, and ending with hybrid_2, which is the simulated experiment.

5.2.3 hybrid_0:

1. On input the security parameter 1^λ and a circuit depth 1^d, the adversary $\mathcal{A}$ outputs $\mathbb{A} \in \mathbb{S}$.
2. Run $\mathsf{Setup}(1^\lambda, 1^d, C, N) \to (pk, sk_1, .., sk_N)$. The adversary is given pk.
3. The adversary outputs a set $S \subseteq \{P_1, \ldots, P_N\}$ such that $S \notin \mathbb{A}$ together with plaintexts $\mu_1, \ldots, \mu_k \in \{0,1\}$. The adversary is handed over $\{sk_i\}_{i \in S}$
4. For each μ_i, the adversary is given $\mathsf{Encrypt}(pk, \mu_i) \to ct_i$.
5. The adversary issues polynomially many queries of the form $(S_i \subseteq \{P_1, \ldots, P_N\}, C_i)$ for circuits $C_i \colon \{0,1\}^k \to \{0,1\} \in \mathcal{C}_\lambda$. After each query the adversary receives for each $l \in S_i$ the value
$$\mathsf{PartDec}(\mathsf{Eval}(C_i, ct_1, \ldots, ct_k), sk_l) \to p_l.$$
6. In the end, $\mathcal{A}$ outputs out. The output of the experiment is out.

5.2.4 hybrid_1:

This hybrid is the same as hybrid_0 except for the following steps.

3. The adversary outputs a set $S \subseteq \{P_1, \ldots, P_N\}$ such that $S \notin \mathbb{A}$ together with plaintexts $\mu_1, \ldots, \mu_k \in \{0,1\}$. The simulator is given the public key, S, and $\mathbb{A}$ and computes
$$\mathsf{Sim.Setup}(pk, S, \mathbb{A}) = (\{sk'_i\}_{i \in S}, \mathsf{state} = (\{sk'_i\}_{i \in S'}, S')).$$
Replacing the simulated secret key shares with the real secret key shares, the adversary is handed over $\{sk_i\}_{i \in S}$ and we set $\mathsf{state} = (\{sk_i\}_{i \in S'}, S'))$.

5 The adversary issues polynomially many queries of the form $(S_i \subseteq \{P_1, \ldots, P_N\}, C_i)$ for circuits $C_i \colon \{0,1\}^k \to \{0,1\} \in \mathcal{C}_\lambda$. After each query, the simulator Sim computes

$$\mathsf{Sim.Query}(C, \{ct_l\}_{l=1}^k, C_i(\mu_1, .., \mu_k), S_i, \mathsf{state}) \to \{p_l\}_{l \in S_i}$$

and sends $\{p_l\}_{l \in S_i}$ to $\mathcal{A}$.

5.2.5 hybrid_2:

3 The adversary outputs a set $S \subseteq \{P_1, \ldots, P_N\}$ such that $S \notin \mathbb{A}$ together with plaintexts $\mu_1, \ldots, \mu_k \in \{0,1\}$. The simulator is given the public key, S, and $\mathbb{A}$ and computes

$$\mathsf{Sim.Setup}(pk, S, \mathbb{A}) = (\{sk'_i\}_{i \in S}, \mathsf{state} = (\{sk'_i\}_{i \in S'}, S')).$$

The adversary is handed over $\{sk'_i\}_{i \in S}$ and we initialize Sim with state $\mathsf{state} = \{sk'_i\}_{i \in S'}, S')$.

We see that $\mathsf{hybrid}_0 = \mathsf{Expt}_{\mathcal{A},Real}(1^\lambda, 1^d)$ and $\mathsf{hybrid}_2 = \mathsf{Expt}_{\mathcal{A},\mathsf{Sim}}(1^\lambda, 1^d)$ and thus, the result follows by Lemmas 6 and 7. □

Lemma 6. *Assuming $B/B_{sm} < \mathsf{negl}(\lambda)$ for some negligible function negl, the statistical distance between the outputs of experiments hybrid_0 and hybrid_1 is negligible in the security parameter, λ.*

Proof. The only change in the view of the adversary $\mathcal{A}$ from hybrid_0 to hybrid_1 is that the partial decryptions p_l are generated by $\mathsf{Sim.Query}$ using a maximal invalid share set of the secret key shares rather than by the real $\mathsf{PartDec}$ with access to the real secret key shares. Let us consider the statistical difference between the two cases. Suppose that the adversary makes m queries and in round $i \in [m]$, the adversary queries the circuit C_i such that $C_i(\mu_1, \ldots, \mu_k) = \mu^i$ and requests the set of partial decryptions $S_i \subseteq \{P_1, \ldots, P_N\}$.

For $l \in S'$, the values $p'_{i,l}$ and $\tilde{p}_{i,l}$ of $\mathsf{PartDec}$ of hybrid_0 and $\mathsf{Sim.Query}$ of hybrid_1, respectively, are both equal to $p'_{i,l} = \tilde{p}_{i,l} = \mathsf{FHE.Decode}_0(\mathsf{Eval}(C_i, ct_1, \ldots, ct_k), sk_l)$ since Sim is given the real secret key shares sk_l for $l \in S'$.

Now, for $l \in S_i \backslash S'$, $\mathsf{PartDec}$ of hybrid_0 generates $p'_{i,l} = \mathsf{FHE.Decode}_0(\mathsf{Eval}(C_i, ct_1, \ldots, ct_k), sk_l)$. By the properties of $\{0,1\}$-LSSS schemes, there exists a minimal valid share set $T_{i,l} \subseteq S' \cup \{l\}$ which contains l and such that

$$\sum_{j \in T_{i,l}} sk_j = sk.$$

Thus,

$$\begin{aligned}\sum_{j\in T_{i,l}} p'_{i,j} &= \sum_{j\in T_{i,l}} \mathsf{FHE.Decode}_0(\mathsf{Eval}(C_i, ct_1, \ldots, ct_k), sk_j)\\ &= \mathsf{FHE.Decode}_0(\mathsf{Eval}(C_i, ct_1, \ldots, ct_k), sk)\\ &= \mu\left\lceil\frac{q}{2}\right\rfloor + e_i\end{aligned}$$

for some noise $e_i \in [-B, B]$ according to the FHE scheme. Rearranging, this yields that in hybrid_0, we have

$$p'_{i,l} = \mu\left\lceil\frac{q}{2}\right\rfloor + e_i - \sum_{j\in T_{i,l}\setminus\{l\}} p'_{i,j}.$$

In hybrid_1, the simulator Sim instead computes $\tilde{p}_{i,l} = \mu\left\lceil\frac{q}{2}\right\rfloor - \sum_{j\in T_{i,l}\setminus\{l\}} \tilde{p}_{i,j}$, which must satisfy $p'_{i,l} - \tilde{p}_{i,l} = e_i$ since $p'_{i,j} = \tilde{p}_{i,j}$ for $j \in S'$. In hybrid_b, the value $e^{sm}_{i,l,b} \xleftarrow{\$} \chi^{sm}_{B_{sm}}$ is sampled and the the adversary receives the value

$$\begin{aligned}\mathsf{hybrid}_0 &: p_{i,l} = p'_{i,l} + e^{sm}_{i,l,0}\\ \mathsf{hybrid}_1 &: p_{i,l} = \tilde{p}_{i,l} + e^{sm}_{i,j,1}.\end{aligned}$$

Since each $e^{sm}_{i,j,b}$ is sampled independently and for each i and l, $p'_{i,l} - \tilde{p}_{i,l} \in [-B, B]$, it follows directly from Lemma 3 and Proposition 1 that the statistical distance between the joint distribution of queries by the adversary and partial decryptions sent back to the adversary in hybrid_0 and hybrid_1 is $\leq mNB/B_{sm} < mN\mathsf{negl}(\lambda)$. Thus, the statistical distance between the outputs the adversary supplies at the end of each hybrid must be negligible in λ. □

Lemma 7. *The outputs of experiments* hybrid_1 *and* hybrid_2 *have the same distribution.*

Proof. The only difference between the two experiments lies in whether the simulator Sim is initialized with state $\mathsf{state}_0 = (\{sk_i\}_{i\in S'}, S')$ or $\mathsf{state}_0 = (\{sk'_i\}_{i\in S'}, S')$. Since any secret is information theoretically hidden by any LSSS scheme, the distributions of a secret sharing of the secret key sk and the secret sharing of the zero-vector $\vec{0}$ are identical as long as the generated shares form an invalid share set. Thus, the distribution of state_0 is identical in the two cases and the conclusion follows. □

5.3 Instantiation

For the security proof to go through we require that $B/B_{sm} < \mathsf{negl}(\lambda)$. For correctness to hold, we require that $B + NB_{sm} < q/4$. This can be achieved by setting $B/B_{sm} < \lambda^{-\log_2 \lambda}$ and $B_{sm} < q/8N$. Such an FHE scheme can be instantiated using known constructions satisfying the properties described in Section 3.1 assuming a variant of Learning with Errors (LWE) assumption. This is as hard as approximating the shortest vector with sub-exponential approximation factors.

6 Construction for t-out-of n Structures

In this section we describe how to construct TFHE for t-out-of-n from the TFHE for $\{0,1\}$-LSSS constructed in Section 5. Essentially, this is done by showing that the class of access structures t-out-of-n is contained in $\{0,1\}$-LSSSD and that finding a $\{0,1\}$-LSSSD share matrix for each such access structure $\mathbb{A} \in t$-out-of-n can be done efficiently. This is done in four steps.

1. We describe a monotone boolean formula with multiple input fan-out computing majority (the $\lceil N/2 \rceil$-out-of-N access structure). This uses the work of [Val84, Gol14] and is described in Section 6.1.

2. Given this boolean formula, we describe a $\{0,1\}$-LSSS matrix M for a related boolean formula with single input fan-out. This uses the 'folklore' algorithm to go from formulas to LSSS matrices and is described in Section 6.2.

3. Using step 2, we construct a TFHE scheme for monotone boolean formulas. From this, we construct TFHE for the majority access structure and the t-out-of-n access structure. This is described in Section 6.3.

6.1 From Majority to Special Monotone Formula

A monotone boolean formula $C\colon \{0,1\}^N \to \{0,1\}$ is a boolean circuit with the following properties.

- There is a single output.
- Every gate is either an AND or an OR gate with fan-in 2 and fan-out 1.
- The input wires can have multiple fan-out.

We define a *special* monotone boolean formula to be a monotone formula where every input has fan-out 1.

First, note that such a boolean formula $C\colon \{0,1\}^N \to \{0,1\}$ defines an access structure in the following way. Let $P = \{P_1, \ldots, P_N\}$ be parties and define the access structure $\mathbb{A}$ such that for any $S \subseteq P$, $S \in \mathbb{A}$ if and only if $C(X_S) = 1$ where $X_S = (X_1, \ldots, X_N) \in \{0,1\}^N$ satisfies that $X_i = 1$ if and only if $P_i \in S$ as described in Notation 3.

Second, from [Val84] there exists a monotone boolean formula for majority. I.e. for every $N \in \mathbb{N}$ there exists a monotone boolean formula $C_{maj}\colon \{0,1\}^N \to \{0,1\}$ such that for $X \in \{0,1\}^N$, $C(X) = 1$ if and only if $\geq \left\lceil \frac{N}{2} \right\rceil$ bits of X are 1. The size of this formula is bounded by $N\ell$ for some $\ell = \mathsf{poly}(N)$. We would like to construct a $\{0,1\}$-LSSSD scheme from C_{maj}, but since it is not obvious how to directly do this, we instead let our construction go through a special monotone boolean formula.

Assume without loss of generality that every input of the monotone boolean formula has fan-out ℓ for an ℓ polynomial in N. Now, define the special monotone boolean formula $C'_{maj} : \{0,1\}^{\ell N} \to \{0,1\}$ as the circuit where every fan-out of an input gate of C_{maj} is its own input in C'_{maj}. More precisely, C'_{maj} is the circuit that can be used to compute majority of $X \in \{0,1\}^N$ in the following way. Given the input tuple $X = (X_1, .., X_N) \in \{0,1\}^N$ for which the majority is to be computed, we first construct a tuple $X' = (X_1, .., X_1, X_2 ....., X_N, .., X_N)$ (i.e. each X_i appears ℓ times). We then compute and output $C'_{maj}(X') = C_{maj}(X)$.

6.2 Constructing $\{0,1\}$-LSSS Matrices from Special Monotone Formulas

In this section we describe a construction that given a special monotone boolean formula $C\colon \{0,1\}^N \to \{0,1\}$ inducing the access structure $\mathbb{A}$ yields an LSSS share matrix M realizing the access structure $\mathbb{A}$. To generate the share matrix M, we rely on a "folklore" algorithm as it is described in [LW11]. We claim that the share matrix M generated by this algorithm satisfies the properties of $\{0,1\}$-LSSS. More specifically, we claim that for shares generated by M and for every valid share set, the secret can be recovered with coefficients in $\{0,1\}$.

We now describe the algorithm formally and prove the claim. Consider the special monotone boolean formula $C\colon \{0,1\}^N \to \{0,1\}$ as a binary tree where the interior nodes are either AND and OR gates and the leaf nodes correspond to inputs. Recall that $(1, .., 0)$ is the target vector for the LSSS scheme. Now, the algorithm is specified in Figure 1.

Theorem 3. *The algorithm described in Figure 1 takes as input a special monotone formula $C : \{0,1\}^N \to \{0,1\}$ and outputs a matrix $M \in \{0,1\}$-LSSS.*

Proof. It is easy to see that the algorithm outputs an LSSS_N share matrix M realizing the access structure $\mathbb{A}$ induced by C. Let the set of parties of $\mathbb{A}$ be $P = \{P_1, \ldots, P_N\}$. We will now show that the share matrix M actually satisfies the properties of $\{0,1\}$-LSSS completing the proof.

Note that any secret sharing using the matrix M gives a single element $w_i \in \mathbb{Z}_p$ to each party P_i. So all that must be shown is that for any secret s shared into parts $s_1, \ldots, s_N$ and any $S \subseteq P$ such that $C(X_S) = 1$ (recalling again Notation 3) or equivalently $S \in \mathbb{A}$ there exists an $S' \subseteq S$ such that $S' \in \mathbb{A}$ and $s = \sum_{i \in S'} s_i$. Viewing C as a binary tree T as in the above algorithm, let v_a be the vector of the label of node a in T and identify the party P_i with the node corresponding to its input in C. We will denote by v_{P_i} the vector of this node, which one can recall is the same vector as the row of the share matrix M that generates the secret share for P_i. We will now show by induction on the height of T that for a minimal valid share set $S \subseteq P$, $\sum_{i \in S} v_{P_i} = v_r$, where r is the root of T. From this the conclusion follows as the root is labelled by $(1, 0, \ldots, 0)$ in the algorithm and that is our target vector.

"Folklore" Algorithm

Input: A special monotone boolean formula $C : \{0,1\}^n \to \{0,1\}$.
Output: An LSSS share matrix M for the access structure induced by C.

1. Label the root node of the tree with the vector (1) (a vector of length one).
2. Initialize a counter count = 1. Start labelling each node m of the tree starting from the top going down.
3. For every non-leaf node m of the tree:
   (a) If m is an OR gate, label its children the same as m.
   (b) If m is an AND gate labeled with the vector v, pad v with $0's$ at the end (if necessary) to make it of length count. Denote the new vector by v'. Then label one of its children with the vector $(v', 1)$ and the other with the vector $(0, \ldots, 0, -1)$ of length count + 1. Then increase count by one.
4. Once the entire tree is labelled, the vectors labelling the leaf nodes form the rows of the LSSS share matrix M. If these vectors have different lengths, the shorter ones are padded with 0s at the end to arrive at vectors of the same length.

Figure 1: Algorithm to convert a formula to LSSS matrix

First, this certainly holds when T has height 1 and 2, i.e. the cases where C is simply an input or the case when C consists of two inputs and a gate.

Second, suppose that the statement holds for every T of height strictly less than $h \geq 2$ and let C be a simple monotone boolean formula such that its binary tree representation T has height h. Denote the root of T by r and let s and t be the children of r. Further, denote by C_s and C_t the circuits induced by the subtree of T with root s and t, respectively, let $\mathbb{A}_s$ and $\mathbb{A}_t$ be the access structures induced by C_s and C_t, respectively, and assume without loss of generality that the leafs of C_s and C_t correspond to parties $P_s = \{P_1, \ldots, P_k\}$ and $P_t = \{P_{k+1}, \ldots, P_N\}$, respectively. Now, we have two cases. If r is an AND gate then for any minimal valid share set $S \subseteq P$ for $\mathbb{A}$, we must have that $S \cap P_s$ and $S \cap P_t$ are minimal valid share sets for $\mathbb{A}_s$ and $\mathbb{A}_t$, respectively, simply by minimality of S. Thus, since by the algorithm $v_r = v_s + v_t$, it follows by the induction hypothesis that

$$\sum_{i \in S} v_i = \sum_{i \in S \cap P_s} v_i + \sum_{i \in S \cap P_t} v_i = v_r.$$

If instead r is an OR gate then for any minimal valid share set $S \subseteq P$ we must have, again by minimality of S, that either $S \cap P_s$ is a minimal valid share set for $\mathbb{A}_s$ and $S \cap P_t = \emptyset$

or vice versa. Since $v_r = v_s = v_t$ by the algorithm, it follows that $\sum_{i \in S} v_i = v_r$. Thus, the statement holds for C, and the induction is complete. □

6.3 Achieving Majority

First, building on Theorem 3 let us state an easy lemma.

Lemma 8. *The* LSSS *share matrix M generated from the algorithm of Figure 1 is a* $\{0,1\}$-LSSS^q *share matrix for any prime q.*

Proof. This follows easily by inspection since the entries of M lie in $\{-1, 0, 1\}$ and the recovery coefficients of M (when viewed as a $\{0,1\}$-LSSS share matrix) lie in $\{0,1\}$. □

Second, an obvious corollary of Lemma 8, Theorem 3, and Proposition 2 is the following theorem.

Theorem 4. *Given a* TFHE *for* $\{0,1\}$-LSSS^q *for some prime q, there exists a* TFHE *scheme for the class of monotone boolean formulas.*

Proof. Let $C\colon \{0,1\}^N \to \{0,1\}$ be a monotone boolean formula and assume without loss of generality that every input of C has fan-out ℓ, polynomial in N. Then let $C'\colon \{0,1\}^{\ell N} \to \{0,1\}$ be the circuit derived from C by letting every fan-out of an input gate of C be its own input in C', just like for C_{maj} and C'_{maj}. By Theorem 3 there is a $\{0,1\}$-LSSS share matrix M for the access structure induced by C' and by allowing collusion between the parties corresponding to the inputs of C' that originally came from the same input gate of C, we get TFHE for the access structure induced by C by Proposition 2. □

Finally, we can construct TFHE for majority as a corollary from C_{maj}.

Corollary 2. *Assuming* TFHE *for* $\{0,1\}$-LSSS^q *for some prime q we have* TFHE *for the majority access structure.*

In [Val84] it is also described how to construct a monotone boolean formula for any threshold t given a monotone boolean formula for majority, C_{maj}.

Corollary 3. *Assuming* TFHE *for* $\{0,1\}$-LSSS^q *for some prime q we have* TFHE *for the t-out-of-n access structure for any t.*

7 Function Secret Sharing from TFHE

In this section we give our construction for (leveled) FSS for any access structure class $\mathbb{S}$ from a TFHE construction for the same access structure class. The syntax and security notion is a natural generalization of the definition given in [BGI15] to arbitrary access structure.

7.1 Definition

The study of functional secret sharing was first initiated in [BGI15, BGI16]. In their definition, reconstruction is simply an addition of partial evaluations in a fixed group. We study a relaxed variant where the complexity of the reconstruction function is allowed to depend on the depth of the function (but is independent of the size of the circuit). The definition below is a natural generalization of the definition in [BGI15] to arbitrary access structures.

Definition 15 (FSS)**.** *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties, let $\mathbb{S}$ be a class of efficient access structure on P, and let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of circuit classes. A function secret sharing scheme for $\mathbb{S}$ and $\mathcal{C}$ is a tuple of PPT algorithms.*

$$\mathsf{FSS} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Decode})$$

satisfying the following specifications:

$(k_1, .., k_N) \leftarrow \mathsf{Gen}(1^\lambda, f, \mathbb{A})$**:** *Takes as input a security parameter λ, a circuit $f : \{0,1\}^k \to \{0,1\} \in \mathcal{C}_\lambda$, and an access structure $\mathbb{A} \in \mathbb{S}$. Outputs function shares $k_1, \ldots, k_N$.*

$p_i \leftarrow \mathsf{Eval}(k_i, x)$**:** *Takes as input a function share k_i and an input $x \in \{0,1\}^k$. Deterministically outputs a partial evaluation p_i.*

$\widehat{\mu} \leftarrow \mathsf{Decode}(B)$**:** *Takes as input a set $B = \{p_i\}_{i \in S}$ for some $S \subseteq \{P_1, \ldots, P_N\}$ where we recall that we identify a party P_i with its index i. Deterministically outputs an evaluation $\widehat{\mu} \in \{0, 1, \perp\}$.*

We ask of FSS *that for any circuit $f : \{0,1\}^k \to \{0,1\} \in \mathcal{C}_\lambda$, parameters $(k_1, .., k_N) \leftarrow \mathsf{Gen}(1^\lambda, f, \mathbb{A})$, any subset $S \subseteq \{P_1, \ldots, P_N\}$, and any $x \in \{0,1\}^k$ the following is satisfied.*

Correctness of Evaluation. *Let $B = \{p_i = \mathsf{Eval}(k_i, x)\}_{i \in S}$. With all but negligible probability in λ over the coins of* Gen*,*

$$\mathsf{Decode}(B) = \begin{cases} f(x), & S \in \mathbb{A} \\ \perp & S \notin \mathbb{A}. \end{cases}$$

Compactness of Evaluation. *There exists a polynomial,* poly*, such that $|p_i| \leq \mathsf{poly}(\lambda, d)$ for any evaluation p_i generated from the* Eval *algorithm of* FSS*, where d is the depth of the circuit f.*[4]

[4]Note that in function secret sharing as proposed in [BGI15], the reconstruction function computes an addition of the partial evaluations, which are of a fixed size. Here we allow the size of the shares and the running time of the reconstruction to grow with the depth, but not the size, of the evaluated circuit.

Security. *Any PPT adversary $\mathcal{A}$ has only negligible advantage as a function of λ over the coins of all the algorithms in the following game:*

1. *On input the security parameter 1^λ, the adversary $\mathcal{A}$ outputs $\mathbb{A} \in \mathbb{S}$ along with the functions (f_0, f_1) of same depth d and size s.*
2. *Sample a random bit $b \in \{0,1\}$. Run $\mathsf{Gen}(1^\lambda, f_b, \mathbb{A}) \to (k_1, .., k_N)$.*
3. *The adversary $\mathcal{A}$ outputs a set $S \subseteq P$ such that $S \notin \mathbb{A}$ and S is a maximal invalid set.*
4. *The adversary receives $\{k_i\}_{i \in S}$.*
5. *The adversary $\mathcal{A}$ issues polynomially many queries. A query is an input x_j satisfying $f_0(x_j) = f_1(x_j)$. After each query the adversary receives the value*

$$\mathsf{Eval}(k_l, x_j) \to p_l$$

   for each $l \in [N]$.
6. *The adversary $\mathcal{A}$ outputs b' and wins if $b = b'$.*

7.2 Construction

Let TFHE be a threshold fully homomorphic encryption scheme for a class of access structures, $\mathbb{S}$. In this section we construct FSS for $\mathbb{S}$ from TFHE.

Construction 2 (FSS)**.** *Let $P = \{P_1, \ldots, P_N\}$ be parties and $\mathbb{S}$ a class of access structures on P. Further, let TFHE be a threshold fully homomorphic encryption scheme for $\mathbb{S}$, let $len(\lambda)$ be the bit-length of the randomness used by $\mathsf{TFHE.PartDec}$, and let a publicly known pseudorandom function $g\colon \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^{len(\lambda)}$ be given. Then define the following algorithms*

$$\mathsf{FSS} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Decode})$$

as follows:

$(k_1, .., k_N) \leftarrow \mathsf{Gen}(1^\lambda, f, \mathbb{A})$**:** *Given the circuit $f\colon \{0,1\}^k \to \{0,1\}$ of size s, let U be the universal circuit accepting a circuit of size s and with k input bits. Let d be the depth of the universal circuit required to run f. Run $\mathsf{TFHE}(1^\lambda, 1^d, \mathbb{A}) \to (pk, sk_1, \ldots, sk_N)$. For each party P_i, compute $k_i = (\mathsf{TFHE.Encrypt}(pk, f), sk_i, r_i)$ for some independently random seed r_i where the encryption is performed bitwise. Output $\{k_i\}_{i=1}^N$.*

$p_i \leftarrow \mathsf{Eval}(k_i, x)$**:** *Parse k_i as $k_i = (ct_f, sk_i, r_i)$. Determine the universal circuit U_x which evaluates a circuit on x. Run $\mathsf{TFHE.Eval}(U_x, ct_f) \to \widehat{ct}_{eval}$. Output*

$$\mathsf{TFHE.PartDec}(ct_{eval}; g(r_i, x)) \to p_i,$$

where $g(r_i, x)$ is the randomness used by $\mathsf{TFHE.PartDec}$.

$\widehat{\mu} \leftarrow \mathsf{Decode}(B)$**:** *Output $\mathsf{TFHE.FinDec}(B) \to \widehat{\mu}$.*

7.3 Analysis

In the following we analyze our FSS scheme. Most of the requirements are almost automatically satisfied by the properties of the TFHE scheme.

Correctness of Evaluation Follows immediately from the correctness of evaluation of the TFHE scheme.

Compactness of the evaluation Follows immediately from the compactness of the TFHE scheme and the compactness of the universal circuit.

Security In our construction, the security game of FSS is almost equivalent to the IND-security game of the TFHE scheme. The only difference is that the randomness of TFHE.PartDec is generated by PRF, but by the security of PRF this means that the two games are indistinguishable and security follows.

8 Distributed Pseudorandom Functions

The syntax and security notion of a DPRF scheme for any access structure class $\mathbb{S}$ is a natural generalization of the definition given in [BLMR13] (which was proposed for threshold access structures).

8.1 Defintion

In this section, we define distributed PRFs for a class of access structures $\mathbb{S}$. We consider DPRF with one bit output. Both the definition and the construction can be generalized for multi-bit outputs. We adapt definitions suitably from [BLMR13].

Definition 16. *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties, let $\mathbb{S}$ be a class of efficient access structure on P, and let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of circuit classes. A distributed PRF scheme* DPRF *for $\mathbb{S}$ is a tuple of PPT algorithms.*

$(k, k_1, .., k_N) \leftarrow \mathsf{Gen}(1^\lambda, \mathbb{A})$**:** *Takes as input a security parameter λ, and an access structure $\mathbb{A} \in \mathbb{S}$. Outputs shares $k_1, \ldots, k_N$ and a key k.*

$p_i \leftarrow \mathsf{Eval}(k_i, x)$**:** *Takes as input a function share k_i and an input $x \in \{0,1\}^\lambda$. Deterministically outputs a partial evaluation p_i.*

$v \leftarrow \mathsf{Decode}(B)$**:** *Takes as input a set $B = \{p_i\}_{i\in S}$ for some $S \subseteq \{P_1, \ldots, P_N\}$ where we recall that we identify a party P_i with its index i. Deterministically outputs an evaluation $v \in \{0, 1, \} \cup \perp$.*

We ask of DPRF *that for any $(k, k_1, .., k_N) \leftarrow \mathsf{Gen}(1^\lambda, \mathbb{A})$, any subset $S \subseteq \{P_1, \ldots, P_N\}$, and any $x \in \{0,1\}^\lambda$ the following is satisfied. Here, g denotes a publicly known pseudorandom function.*

Correctness of Evaluation. *Let* $B = \{p_i = \mathsf{Eval}(k_i, x)\}_{i \in S}$. *With all but negligible probability in* λ *over the coins of* Gen,

$$\mathsf{Decode}(B) = \begin{cases} g(k, x), & S \in \mathbb{A} \\ \perp & S \notin \mathbb{A}. \end{cases}$$

Security. *Any PPT adversary* $\mathcal{A}$ *has only negligible advantage as a function of* λ *over the coins of all the algorithms in the following game:*

1. *On input the security parameter* 1^λ, *the adversary* $\mathcal{A}$ *outputs* $\mathbb{A} \in \mathbb{S}$ *and a set* $S \notin \mathbb{A}$.
2. *Run* $\mathsf{Gen}(1^\lambda, \mathbb{A}) \to (k, k_1, .., k_N)$.
3. *The adversary* $\mathcal{A}$ *receives* $\{k_i\}_{i \in S}$.
4. *The adversary* $\mathcal{A}$ *issues polynomially many distinct queries of the form* $x_j \in \{0,1\}^\lambda$ *for* $j \in [q]$. *The adversary receives* $\{\mathsf{Eval}(k_i, x_j)\}_{i \in [N] \setminus S}$.
5. *Finally adversary outputs* x^*, S^* *which is not equal to any of the queries above.* $S^* \cup S \notin \mathbb{A}$.
6. *The challenger samples* $b \xleftarrow{\$} \{0,1\}$. *If* $b = 0$, *set* $v = g(k, x^*)$ *otherwise sample* $v \xleftarrow{\$} \{0,1\}$. *Adversary is given* $\{\mathsf{Eval}(k_i, x^*)\}_{i \in S^* \setminus S}$ *along with* v.
7. *The adversary* $\mathcal{A}$ *continues to issue polynomially many distinct queries of the form* $x_j \neq x^*$ *for* $j \in [q]$. *The adversary receives* $\{\mathsf{Eval}(k_i, x_j)\}_{i \in [N] \setminus S}$.
8. *The adversary* $\mathcal{A}$ *outputs* $b' \in \{0,1\}$ *and wins if* $b' = b$.

The advantage of any adversary is defined as $| Pr[b' = b] - 1/2 |$

Remark. We also define selective security where the adversary is not allowed to ask for the set S^*. Any selectively secure scheme can be converted to an adaptively secure scheme assuming subexponential hardness assumptions by using complexity leveraging (by guessing S^*).

8.2 Construction

Theorem 5. *Assuming a secure* TFHE *scheme for a class of access structures* $\mathbb{S}$ *and polynomial sized circuits, there exists a selectively secure* DPRF *scheme for* $\mathbb{S}$.

We now present our construction below for PRF with one-bit output. The construction can be generalised for multi-bit output.

Construction 3 (DPRF)**.** *Let* $P = \{P_1, \ldots, P_N\}$ *be parties and* $\mathbb{S}$ *a class of access structures on* P. *Further, let* TFHE *be a threshold fully homomorphic encryption scheme for* $\mathbb{S}$. *Then define the following algorithms:*

$$\mathsf{DPRF} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Decode})$$

as follows:

$(k, k_1, .., k_N) \leftarrow \mathsf{Gen}(1^\lambda, \mathbb{A})$**:** *Let g denote a pseudo-random function $g : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}$. Let g_{len} denote the PRF with output of len bits. Let d denote the depth of this function $g()$ represented as a circuit. Sample $k \xleftarrow{\$} \{0,1\}^\lambda$. Run $\mathsf{TFHE}(1^\lambda, 1^d, \mathbb{A}) \to (pk, sk_1, \ldots, sk_N)$. For each party P_i, compute $k_i = \mathsf{TFHE.Encrypt}(pk, k), sk_i, r_i)$ for $r_i \xleftarrow{\$} \{0,1\}^\lambda$ where the encryption is performed bitwise. Output $(k, k_1, .., k_N)$.*

$p_i \leftarrow \mathsf{Eval}(k_i, x)$**:** *Parse k_i as $k_i = (ct, sk_i, r_i)$. Run $ct_x \leftarrow \mathsf{TFHE.Eval}(g(\cdot, x), ct)$. Then compute and output $p_i \leftarrow \mathsf{TFHE.PartDec}(sk_i, ct_x; g_{len}(r_i, x))$. Here len is the length of the randomness required by the algorithm* PartDec*.*

$v \leftarrow \mathsf{Decode}(B)$**:** *Takes as input a set $B = \{p_i\}_{i \in S}$ for some $S \subseteq \{P_1, \ldots, P_N\}$ where we recall that we identify a party P_i with its index i. Run and output $v = \mathsf{TFHE.FinDec}(B)$*

Correctness follows from the correctness of the TFHE scheme. The proof of security for our construction of a distributed PRF is as follows.

8.2.1 Security Proof.

We now describe hybrids where the first hybrid (hybrid_0) corresponds to the distributed PRF security game with $b = 0$ and the last hybrid is independent of the prf key k (with advantage 0 in this hybrid). Then we show why each hybrid is indistinguishable.

- hybrid_0 : This hybrid corresponds to the selective distributed PRF security game with $b = 0$.
- hybrid_1 : This hybrid is the same as the previous hybrid except that while responding to the queries of the adversary, for all parties P_i, $i \in [N] \setminus S$, for every query x (including the challenge) partial decryption is computed using true randomness instead of $g_{len}(r_i, x)$.

  The two hybrids described above are indistinguishable due to the security of the PRF g_{len}.
- hybrid_2 : This hybrid is the same as the previous hybrid except that the queries of the adversary are simulated using the simulator of the TFHE scheme. For every x, set $v_x = g(k, x)$ and use this to simulate the TFHE partial decryptions computed to respond to the queries.

  The two hybrids described above are indistinguishable due to simulation security of TFHE.

- hybrid_3 : This hybrid is the same as the previous hybrid except that ct is generated as an encryption of 0^λ.

  The two hybrids described above are indistinguishable due to semantic security of TFHE.

- hybrid_4 : This hybrid is the same as the previous hybrid except that the for the challenge input x^*, $v \xleftarrow{\$} \{0,1\}$ is given out.

  The two hybrids described above are indistinguishable due to security of the PRF g

- hybrid_5 : This hybrid is the same as the previous hybrid except that ct is generated as an encryption of the key k.

  The two hybrids described above are indistinguishable due to semantic security of TFHE.

- hybrid_6 : This hybrid is the same as the previous hybrid except that the queries of the adversary are computed using partial decryption algorithm of the TFHE scheme with $g_{len}(r_i, x)$ as the randomness to compute partial decryption for every party i and input x. This corresponds to the security game with $b = 1$

  The two hybrids described above are indistinguishable due to security of the PRF g_{len}

References

[AJL+12] 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 *EUROCRYPT*, pages 483–501, 2012.

[AJN+16] Prabhanjan Ananth, Aayush Jain, Moni Naor, Amit Sahai, and Eylon Yogev. Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption. In *CRYPTO*, pages 491–520, 2016.

[Bei96] Amos Beimel. Phd thesis. *Israel Institute of Technology, Technion, Haifa, Israel,*, 1996.

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *EUROCRYPT*, 2015.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In *CRYPTO*, pages 509–539, 2016.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *CRYPTO*, pages 410–428, 2013.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *CRYPTO*, pages 93–122, 2016.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GLS15] S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In *CRYPTO*, pages 63–82, 2015.

[Gol14] Oded Goldreich. Valiant's polynomial-size monotone formula for majority. 2014.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013.

[KNY16] Ilan Komargodski, Moni Naor, and Eylon Yogev. How to share a secret, infinitely. In *TCC*, pages 485–514, 2016.

[KRR14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *STOC*, pages 485–494, 2014.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, 2012.

[LW11] Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, 2011.

[MS95] Silvio Micali and Ray Sidney. A simple method for generating and sharing pseudo-random functions, with applications to clipper-like escrow systems. In *CRYPTO*, pages 185–196, 1995.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round MPC from LWE via multi-key FHE. In *EUROCRYPT*. 2016.

[Nie02] Jesper Buus Nielsen. A threshold pseudorandom function construction and its applications. In *CRYPTO*, pages 401–416, 2002.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdcs. In *EUROCRYPT*, pages 327–346, 1999.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004.

[Val84] Leslie G. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 1984.