Efficient Threshold FHE with Application to Real-Time Systems

Siddhartha Chowdhury[1], Sayani Sinha[1], Animesh Singh[1], Shubham Mishra[2], Chandan Chaudhary[1], Sikhar Patranabis[3], Pratyay Mukherjee[4], Ayantika Chatterjee[1], and Debdeep Mukhopadhyay[1]

[1]IIT Kharagpur
[2]UC Berkeley
[3]IBM Research, India
[4]SupraOracles Research

June 1, 2023

Abstract

Threshold Fully Homomorphic Encryption (ThFHE) enables arbitrary computation over encrypted data while keeping the decryption key distributed across multiple parties at all times. ThFHE is a key enabler for threshold cryptography and, more generally, secure distributed computing. Existing ThFHE schemes inherently require highly inefficient parameters and are unsuitable for practical deployment. In this paper, we take the first step towards making ThFHE practically usable by (i) proposing a novel ThFHE scheme with a new analysis resulting in significantly improved parameters; (ii) and providing the first practical ThFHE implementation benchmark based on Torus FHE.

- We propose the *first practical* ThFHE scheme with a *polynomial modulus-to-noise ratio* that supports practically efficient parameters while retaining provable security based on standard quantum-safe assumptions. We achieve this via a novel Rényi divergence-based security analysis of our proposed threshold decryption mechanism.
- We present an optimized software implementation of a Torus-FHE based instantiation of our proposed ThFHE scheme that builds upon the existing Torus FHE library and supports (distributed) decryption on highly resource-constrained ARM-based handheld devices. Along the way, we implement several extensions to the Torus FHE library, including a Torus-based linear integer secret sharing subroutine to support ThFHE key sharing and distributed decryption for any threshold access structure.

We illustrate the efficacy of our proposal via an end-to-end use case involving encrypted computations over a real medical database, and distributed decryptions of the computed result on resource-constrained ARM-based handheld devices.

Contents

1 Introduction **3**

1.1 Our Contribution 4

2 Related Work **5**

3 Preliminaries and Background **8**

3.1 Notations and Mathematical Background 8

3.2 Fully Homomorphic Encryption (FHE) 9

3.3 Threshold FHE 10

3.4 Linear Integer Secret Sharing Scheme (LISSS) 13

4 Our Proposal: Torus-FHE with Threshold Decryption **14**

4.1 Decryption in Torus-FHE 15

4.2 Achieving (T, T)-Distributed Decryption 16

4.3 Extending Benaloh-Leichter LISSS 16

4.4 Generalized (t, T)-Threshold Decryption Protocol 18

4.5 Polynomial Modulus-to-Noise Ratio via Rényi Divergence 19

4.6 Analysis of Generalized (t, T)-threshold FHE 23

4.6.1 Correctness of (t, T)-Threshold Decryption 23

4.6.2 IND-Security of (t, T)-Threshold Decryption 24

5 Software Implementation and Experimental Evaluation **27**

5.1 Implementation Details 28

5.2 Experimental Evaluation 28

6 Case-Study: Computing over Encrypted Medical Data **31**

6.1 Encrypted KNN Computation 31

6.2 Experimental results 32

7 Conclusion and Future Work **33**

A Observing the Pattern of Secret Shares **41**

1 Introduction

Outsourced Computation. The recent advent of cloud computing technologies [Hay08, WVLY+10] enables individuals and organizations to outsource heavy computations over large databases to potentially untrusted third-party servers. However, this poses new challenges for the security and privacy of the data, particularly when the data contains sensitive information such as individual medical records, etc. For compliance, regulation, and other essential privacy requirements, the data must be kept secure at rest, in transit, and during computation.

Fully Homomorphic Encryption (FHE). While traditional encryption procedures are useful for securing data at rest and in transit, they often fail to achieve any security during computation. Fully Homomorphic Encryption (FHE) [Gen09, BGV14, BGG+18] resolves this problem by enabling computation on encrypted data. This motivates a significant body of research work [SS10, CNT12, DM15a, CGGI20, CGBH+18, FSK+21] to focus onto building practically efficient fully homomorphic encryption systems.

Threshold Cryptography. While FHE resolves the crucial problem of computation on encrypted data, one must carefully store the decryption key securely to get any real benefit out of it. Typical enterprise solutions of key management involve using secure hardware solutions such as HSMs, SGXs etc. While they provide reasonable security in practice, they often suffer from a lack of programmability, cumbersome setup procedures, scalability, high cost, side-channel attacks etc [KHF+19, LSG+18]. An alternative approach, that uses threshold cryptography [Sha79, DF90, DDFY94] is offered by enterprises like Hashicorp Vault[1]. In that approach, the key is shared among multiple servers (say T) to avoid a "single point of failure" and a threshold number of them (say t) can collaborate to recompute the decryption key. However, this defies the purpose as a single compromise at the decryption server, during a key-reconstruction, would reveal the key entirely. An ideal solution must have the decryption key distributed *at all time*. In particular, this is achieved by a ThFHE (Threshold-FHE) scheme [AJL+12, MW16, BGG+18, CCK23], where the decryption is performed jointly by any threshold number of parties without reconstructing the key at any one place. In particular, parties compute partial decryption with their shares of the key and send them over to the decryptor, who, once obtains t such partial decryptions in total (may include her partial decryption), combines them to get the message.

Practical ThFHE. While there are several ThFHE schemes in the literature [AJL+12, MW16, BGG+18, MS+11, JRS17, CCK23], the state-of-art is far from being practical. This is in contrast to the literature in FHE, in that many practical proposals and prototypes exist[2]. Perhaps the most crucial bottleneck of the existing schemes comes from the security requirement imposed by the threshold decryption procedure, which might involve up to $t-1$ corrupted servers (we only consider passive/semi-malicious corruption here). In slightly more detail, the modulus to noise ratio used in the existing threshold schemes must be set *super-polynomial* (in the security parameter) compared to the non-threshold FHE schemes that require only a polynomial modulus to noise ratio. The use of super-polynomial modulus-

[1] https://www.vaultproject.io/

[2] https://homenc.github.io/HElib/, https://www.microsoft.com/en-us/research/project/microsoft-seal/, https://tfhe.github.io/tfhe/, https://palisade-crypto.org/

noise ratio stems from a technique called *smudging* (alternatively *noise flooding*), which is used to achieve security when the parties are corrupt during the distributed decryption. In this work, we propose the first practical ThFHE scheme, which uses polynomial modulus to noise ratio – we achieve this by adapting a Rényi divergence-based technique for distinguishing problems with public sampleability property as discussed in [BLRL+18, TT15]. This dramatically improves the system's efficiency, as shown by our prototype implementation in software – this is the first benchmark for a ThFHE scheme.

1.1 Our Contribution

In this work we significantly improve the state-of-art for practical ThFHE scheme by both *new theoretical analysis* and *first prototype implementations*. Finally, we complement this by providing a use case for a real-world, end-to-end system that securely computes on outsourced medical data and distributed decryption of the computed result is performed by distributing the key among different lightweight devices that medical personnel hold while avoiding the single-point of failure.

First Practical Threshold FHE Scheme with Polynomial Modulus to Noise Ratio. Our construction is based on the prior constructions [AJL+12, MW16, CM15]. In particular, we plug-in the threshold decryption technique from Asharov et al. [AJL+12] into the FHE scheme by Gentry, Sahai and Water [GSW13] (GSW) – as a result, we get a *single-key* ThFHE version of the scheme by Mukherjee and Wichs [MW16] with two crucial differences: (i) the smudging noise is sampled from a Gaussian distribution; (ii) a polynomial modulus is used. In our analysis, which is inspired by the works such as [BLRL+18, TT15, ASY22], we use Rényi Divergence instead of statistical distance, which essentially made those changes possible and achieves indistinguishability-based notion of security [JRS17]. As a result, we obtain the *first practical* ThFHE *scheme with polynomial modulus to noise ratio*.[1]

First Software Prototype for Threshold FHE. We provide the first prototype implementation of a ThFHE system with a benchmark in software. We expand further below.

- In our software implementation, we provide an extension of the existing library for Torus-FHE[2]. We also provide the first software implementation of a linear integer secret sharing scheme extended from [DT06] to support Torus Ring-LWE secret key sharing, which may be of independent interest. Our extended Torus-FHE library supports arbitrary t out of T threshold decryption while maintaining polynomial modulus-to-noise ratio.
- To emulate our intended use-case of decryption in handheld devices, we develop a portable implementation of the threshold decryption routines. We provide the results from its experimentation on a Raspberry Pi 3b board that uses a 64 bit ARM CPU.

[1] Remarkably, polynomial modulus to noise ratio not only improves the efficiency significantly, but also makes the scheme potentially more secure – this is because such a ratio for the underlying Learning with Errors problem [Reg09] implies reduction to the corresponding worst-case lattice problem with polynomial approximation factor, which are believed to be significantly harder than the same problem with super-polynomial approximation factor, which is obtained if a super-polynomial ratio is used. For more details, we refer to, for example, [BV14].

[2] https://tfhe.github.io/tfhe/

A Practical Use-case. Finally, as a use-case, we provide a detailed description of an end-to-end secure computation system over outsourced encrypted medical data. The goal is to have encrypted medical data stored in the cloud, such that any heavy computation may be performed on that encrypted data. At the same time, the decryption key must be stored in an easily accessible but secure way. In particular, a medical personnel who owns many lightweight devices should be able to access the result of the computation by using t devices, but if any $t-1$ device are compromised,[1] then the decryption key must not be revealed, even if the compromised device participates in several decryption sessions. For example, in a (5,8)-threshold decryption system, any five devices should be able to perform the distributed decryption, and the decryption key should remain secure until the number of compromised parties is less than five. Furthermore, the system should be such that the encryption or the computation on the encrypted data should be oblivious to the values of t or T. In particular, one may think about changing those values later. Our system satisfies all of the above mentioned aspects.

2 Related Work

Threshold FHE. The concept of ThFHE, introduced by Asharov et al. [AJL$^+$12], has been majorly studied in two related but slightly different contexts: (i) to build low-round multiparty computation protocols [AJL$^+$12, MW16, GLS15, BJMS20]; (ii) and as a key enabler for threshold cryptography [BGG$^+$18, JRS17, CCK23]. At a technical level these two categories of schemes follow slightly different definitions because of different application requirements. The MPC-motivated works (category (i) above) consider mainly (T,T)-threshold settings (Badrinarayan et al. [BJMS20] is an exception), whereas the later works are focused towards achieving (t,T) $(t \leq T)$ setting (which is standard in the threshold cryptography literature). Furthermore, the former works (necessarily, due to requirement of MPC) considered distributed key-generation for single-key schemes,[2] unless, of course, a specialized public-key infrastructure was assumed. The only distributed step considered by the threshold-inspired works (category-(ii) above) was distributed decryption, in that every party has a common ciphertext and their own share of secret decryption key; and then each party broadcasts a partially decrypted ciphertext generated locally, which are then combined together to obtain the decrypted value – this is similar to threshold public-key encryption [BBH06, Fra90, DF90, SG02]. The distributed decryption step is modular and essentially agnostic of how the ciphertext is generated. In particular, such decryption protocol can be plugged-in to schemes with appropriate distributed key-generation protocol or can be used in a multi-key scheme a la [MW16, BJMS20] (or even with a symmetric-key scheme).[3] Therefore, distributed decryption step appears in both categories of the above

[1] In this work we consider a semi-malicious model of corruption a la [AJL$^+$12, MW16] which assumes that corrupt parties behave as per the protocol description except they can choose arbitrary values for randomness – this is stronger than the passive security model where parties choose good randomness but weaker than fully malicious setting where parties behave in completely arbitrarily manner.

[2] The multi-key schemes are the exceptions. For multi-key schemes such as Mukherjee and Wichs [MW16] the key-generation step was naturally dispensed with, which was the key-step to achieve round-optimal MPC in the common random string model.

[3] In a (T,T) setting the distributed key-generation is trivial [AJL$^+$12]. In the (t,T) setting the key shares must be consistent with the secret $(t-1)$ degree polynomial, and hence a non-trivial protocol is required. One may just think about using a generic MPC protocol for this a la [BJMS20]. More efficient protocols have been considered recently [GHL22]. This is not the focus of our work.

work. Our focus here is more aligned with the threshold cryptography literature, and hence we follow the second approach.

One common aspect of all of the above distributed decryption constructions is the use of the so-called noise smudging technique to achieve a simulation-based security guarantee when up to $(t-1)$ parties are (semi-maliciously) corrupt. The main idea is to sample noise from a Gaussian distribution and then use it to "smudge" (alternatively "flood") the "sensitive LWE noise" in the partially decrypted ciphertext. The analysis (based on simple statistical distance measurements) crucially relies on the smudging noise being super-polynomially larger than the LWE noise; then to ensure correctness one must use a super-polynomial modulus-to-noise ratio – this results in impractical parameters. In this paper, we instead use a novel Rényi divergence-based analysis inspired by [BLRL$^+$18, TT15] – this allows us to use a polynomially large smudging noise and subsequently a polynomial modulus-to-noise ratio, thereby putting ThFHE in the practical regime.

Next, we compare our work with two concurrent and independent works [BS23, DWF22] that also aim to design ThFHE schemes with polynomial modulus-to-noise ratio while relying on Renýi divergence-based arguments. In particular, we highlight the key technical differences between our approach and the approaches used in these works, and the resulting differences in terms of security, practical efficiency, and reliance on assumptions (such as random oracles).

Comparison with [BS23]. A concurrent and independent work by Boudgoust and Scholl [BS23] proposes a poly-modulus threshold FHE scheme that achieves full-fledged IND-CPA security with partial decryption query simulatability. They follow a two-step proof strategy: (a) argue one-way CPA security based on Rényi divergence (which, unlike our approach, does not require a public sampleability argument), and (b) use an additional transformation to achieve full-fledged security, which either uses a random oracle (RO) or Goldreich-Levin (GL) hard-core predicates [GL89]. As mentioned in [BS23], the RO-based construction incurs significant limitations in terms of homomorphic computation capabilities (in particular, since the RO itself does not have an efficient circuit description), while the hardcore predicate-based construction incurs additional overheads, particularly due to larger ciphertext size. On the other hand, we prove the indistinguishability-based security of ThFHE using a Rényi divergence-based public sampleability argument in the standard model, thereby avoiding random oracles and preserving the fully homomorphic computation capabilities of the underlying scheme without any additional overheads.

Comparison with [DWF22]. Another concurrent and independent work by Dai et al. [DWF22] proposed ThFHE schemes while relying on Rényi divergence-based arguments. The work proposes two approaches – the first based on leakage-resilient Dual-GSW (DGSW) [BHP17], and the other based on RO. The first approach relies on a security argument that seems to hold only for a single (or at most a constant number of) partial decryption query (queries), and it is unclear how the analysis would extend to polynomially many partial decryption queries (and what the corresponding effect on the scheme's noise parameters would be). On the other hand, our security model allows polynomially many partial decryption queries and we prove the security of our proposed ThFHE scheme in this model, while also formalizing the effect of the number of queries on the noise parameters for our scheme. We believe that for real-world applications, it is reasonable to assume that the adversary is allowed to see polynomially many decryption queries, and any restriction thereof is perhaps undesirable.

Finally, as mentioned earlier, our approach avoids the use of RO (and any restrictions to the homomorphic computation capabilities resulting from such an approach).

In the rest of this section, we mention some additional related work with different goals and/or different security models as compared to our work.

Efficient FHE Bootstrapping. In a recent work, Lee et al. [LMK+22] proposed improved bootstrapping methods for FHEW/Torus FHE [DM15b,CGGI20] and their threshold versions. They do not focus on achieving ThFHE with polynomial modulus-to-noise ratio, which is the main focus of our work. Our techniques are agnostic of the bootstrapping procedure used during homomorphic evaluations (and can be potentially combined with the bootstrapping techniques of [LMK+22] to achieve more efficient ThFHE schemes; we leave this as an interesting open question).

Approximate and Circuit-Private FHE. Some other recent works [LMSS22,KS23] focus on achieving stronger security notions (namely IND-CPA^D security) and circuit-privacy for approximate FHE schemes (e.g., CKKS [CKKS17]) using differential privacy tools. Again, their goals and security models are orthogonal to ours, as we focus on designing efficient threshold decryption mechanisms for *exact* FHE schemes (such as Torus FHE) under a different (and incomparable) security definition as compared to IND-CPA^D. Consequently, the security analysis and lower bounds on parameter choices described in [LMSS22, KS23] are seemingly inapplicable to our scheme and differ conceptually from our Rényi divergence-based security analysis of ThFHE.

Multi-Key FHE. In a multi-key FHE scheme, the parties encrypt their input with individual keys (generated locally) and then broadcast them; subsequently, an extended ciphertext is constructed using all the encryptions from the involved parties, and any arbitrary homomorphic operation can be performed on the extended ciphertext [LATV12, CM15, MW16, BP16, PS16, CZW17, CO17, CCS19, AJJM20]. We do not focus on multi-key FHE in this work; however, as mentioned earlier, our distributed/threshold decryption approach and the corresponding security analyses can be adapted to the multi-key setting. We leave this as an interesting direction of future research.

Multiparty HE. Mouchet et al. [MTBH21] recently considered a new notion of multiparty homomorphic encryption scheme (MPHE), which is very similar to the Asharov's et al. [AJL+12]'s threshold FHE notion, that has both distributed key-generation plus distributed decryption, albeit for a (T, T) access structure. They also included an implementation benchmark [MBTPH20]. A subsequent construction secure against malicious adversary has been proposed [CMS+23] recently. However, a major shortcoming of their definition is the absence of a simulation-based definition for their partial decryption protocol – so it does not capture a realistic threat model where adversary can corrupt parties while participating in the decryption procedure. Therefore, they did not need to use any noise smudging. Therefore, their implementation can not be counted as a predecessor of ours. Another work by Ananth et al. [AJJM20] defines another primitive, which they also call multiparty homomorphic encryption – this is a slightly weaker variant of multi-key FHE, in that the decryption computation complexity grows with the circuit being evaluated. Padron and Vargas [PV21] define an even weaker primitive (where the evaluator holds part of the secret key) and calls it multiparty homomorphic encryption. Our notion of ThFHE and the corresponding security definition differ significantly from all of the above mentioned notions.

Software Frameworks. Recent works have accelerated FHE (non-threshold) implementations via GPU based parallelizations. Based on [CGGI20], a Python library **NuFHE** [1] has been developed. In [CDS15], the **Cingulata** (formerly, Armadillo) C++ toolchain and run-time environment were introduced for running programs over FHE ciphertexts, which now supports Torus FHE. **Lattigo**[2] [MBTPH20] on the other hand is a Go based module that builds secure protocols based on Multiparty-Homomorphic-Encryption and Ring-Learning-With-Errors-based Homomorphic Encryption Primitives. Some recent extensions proposed in [MBH22, MTBH21] do support threshold decryption; however, all of these implementations fundamentally require a superpolynomial modulus-to-noise ratio. Additionally, they only support *leveled* homomorphic versions (i.e., without bootstrapping) of the BGV [BGV14], BFV [Bra12, FV12] and CKKS [CKKS17] FHE schemes. Our ThFHE implementation builds upon and extends the Torus FHE library in a natural way (including the bootstrapping procedure), and is cross-compatible with all of these computation frameworks.

3 Preliminaries and Background

In this section, we introduce the notations used throughout this paper. We also present some preliminary background material on cryptographic primitives used in this paper.

3.1 Notations and Mathematical Background

Notations. We use $\mathbb{T}$ to denote the Torus (i.e., the set of all real numbers modulo 1). We write $x \leftarrow \chi$ to represent that an element x is sampled uniformly at random from a set/distribution $\mathcal{X}$. For $a, b \in \mathbb{Z}$ such that $a, b \geq 0$, we denote by $[a]$ and $[a, b]$ the set of integers lying between 1 and a (both inclusive), and the set of integers lying between a and b (both inclusive). We refer to $\lambda \in \mathbb{N}$ as the security parameter, and denote by $\text{poly}(\lambda)$ and $\text{negl}(\lambda)$ any generic (unspecified) polynomial function and negligible function in λ, respectively.[3]

LWE Assumption and its Variants. Here, we recall the Learning with Errors (LWE) assumption and some of its variants, including the Ring LWE (RLWE) assumption and the Binary RLWE assumption.

LWE Assumption. Let $\lambda \in \mathbb{N}$ be a security parameter, and let $q, n, m = \text{poly}(\lambda)$. For each $i \in [m]$, let

$$\boldsymbol{a}_i \leftarrow \mathbb{Z}_q^n, \quad b_i = \boldsymbol{a_i} \cdot \boldsymbol{s} + e_i, \quad u_i \leftarrow \mathbb{Z}_q,$$

where $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ is a uniformly sampled secret vector, $e_i \leftarrow \psi$ where ψ is a Gaussian noise distribution over $\mathbb{Z}_q$, and $\boldsymbol{a}_i \cdot \boldsymbol{s}$ denotes the vector dot-product between the vectors $\boldsymbol{a}_i$ and $\boldsymbol{s}$. The LWE hardness assumption states that, for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the following holds:

$$|\mathsf{Pr}[\mathcal{A}(\{\boldsymbol{a}_i, b_i\}_{i\in[m]}) = 0] - \mathsf{Pr}[\mathcal{A}(\{\boldsymbol{a}_i, u_i\}_{i\in[m]}) = 0]| < \text{negl}(\lambda).$$

[1] https://nufhe.readthedocs.io/en/latest/
[2] https://github.com/tuneinsight/lattigo
[3] Note that a function $f : \mathbb{N} \to \mathbb{N}$ is said to be negligible in λ if for every positive polynomial p, $f(\lambda) < 1/p(\lambda)$ when λ is sufficiently large.

RLWE Assumption. Let $\lambda \in \mathbb{N}$ be a security parameter, and let $q, N, m = \text{poly}(\lambda)$. For each $i \in [m]$, let

$$A_i(X) \leftarrow \mathbb{Z}_q[X]/(X^N+1), \quad B_i(X) = A_i(X) \cdot S(x) + E_i(X),$$
$$U_i(X) \leftarrow \mathbb{Z}_q[X]/(X^N+1),$$

where $S(X) \leftarrow \mathbb{Z}_q[X]/(X^N+1)$ is a uniformly sampled secret polynomial, $E_i(X) \leftarrow \psi[X]/(X^N+1)$ where ψ is a Gaussian noise distribution over $\mathbb{Z}_q$, and $A_i(X) \cdot S(X)$ denotes the polynomial multiplication modulo (X^N+1) between $A_i(X)$ and $S(X)$. The Ring LWE (RLWE) hardness assumption states that, for any PPT adversary $\mathcal{A}$, we have:

$$\left|\mathsf{Pr}\left[\mathcal{A}\left(\{A_i(X), B_i(X)\}_{i\in[m]}\right) = 0\right] - \mathsf{Pr}\left[\mathcal{A}\left(\{A_i(X), U_i(X)\}_{i\in[m]}\right) = 0\right]\right| < \text{negl}(\lambda).$$

Binary RLWE. The Binary RLWE (BRLWE) hardness assumption is a variant of the RLWE hardness assumption described above where, the secret key polynomial $S(X)$ is sampled from $\mathbb{B}[X]/(X^N+1)$ as opposed to $\mathbb{Z}_q[X]/(X^N+1)$, where $\mathbb{B} = \{0,1\}$. Note that although an equivalence between the LWE with binary secrets assumption and the standard LWE assumption is known [Mic18], a similar result for BRLWE and RLWE is not known to the best of our knowledge. However, the BRLWE hardness assumption is widely believed to hold [BBPS19, BD20].

Threshold Access Structure. For any $T, t \in \mathbb{N}$ such that $t \leq T$, a (t,T)-threshold access structure over any set $\mathcal{P} = \{P_1, \ldots, P_T\}$ is defined as a collection of qualified subsets of the form

$$\mathbb{A}_{(t,T)} = \{\overline{\mathcal{P}} \subseteq \mathcal{P} : \left|\overline{\mathcal{P}}\right| \geq t\},$$

which (informally) states that any subset with t or more parties is a qualified subset. If $\mathbb{A}_{(t,T)}$ is a *minimal* (t,T)-threshold access structure, then it only consists of subsets of size exactly t; in other words, we have $|\mathbb{A}_{(t,T)}| = \binom{T}{t}$. Observe that this access structure can be represented efficiently [BGG+18]. In particular, there exists a polynomial-size circuit that takes as input T-length vectors and outputs a bit, such that for every valid subset $S \in \mathbb{A}_{(t,T)}$, on input the T-sized binary vector $V = \{V_i\}_{i\in[T]}$ with $V_i = 1$ if and only if $P_i \in S$, the circuit outputs 1 (see [BGG+18] for details).

Rényi Divergence. Let $Supp(P)$ and $Supp(Q)$ denote the supports of distributions P and Q respectively, such that $Supp(P) \subseteq Supp(Q)$. For $a \in (1, +\infty)$, the Rényi divergence of order a is

$$R_a(P||Q) = \left(\sum_{x\in Supp(P)} \frac{P(x)^a}{Q(x)^{a-1}}\right)^{\frac{1}{a-1}}.$$

This definition extends naturally to continuous distributions (see [BLRL+18] for details).

3.2 Fully Homomorphic Encryption (FHE)

Fully Homomorphic Encryption (FHE) is a form of encryption that permits computations directly over encrypted data without decrypting it first. The result of this computation is also encrypted. Below, we recall the definition of fully homomorphic encryption (FHE) [Gen09, GHS12] for any message space $\mathcal{M}$.

Definition 1 (Fully Homomorphic Encryption). *A fully homomorphic encryption (FHE) scheme is a tuple of four algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *with respect to a class of Boolean functions* $\mathcal{F} = \{\mathcal{F}_\ell\}_{\ell\in\mathbb{N}}$ *(represented as Boolean circuits with* ℓ*-bit inputs) such that the tuple* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is an IND-CPA-secure public-key encryption (PKE) scheme as defined below, and the evaluation algorithm* Eval *satisfies the homomorphism and compactness properties as defined below:*

IND-CPA security: *For any* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*, for any messages* $\mathsf{m}_0, \mathsf{m}_1 \in \mathcal{M}$*, and for any probabilistic polynomial-time (PPT) adversary* $\mathcal{A}$*, letting* $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_0)$ *and* $\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_1)$*,*

$$|\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{m}_0, \mathsf{m}_1, \mathsf{ct}_0) = 1] - \Pr[\mathcal{A}(\mathsf{pk}, \mathsf{m}_0, \mathsf{m}_1, \mathsf{ct}_1)] = 1| \leq \mathrm{negl}(\lambda).$$

Correctnesss: *The homomorphism of the FHE scheme ensures correctness. For any (Boolean) function* $f : \{0,1\}^\ell \to \{0,1\} \in \mathcal{F}$ *and any sequence of* ℓ *messages* $\mathsf{m}_1, \ldots, \mathsf{m}_\ell$*, letting* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*, and* $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_i)$ *for each* $i \in [\ell]$*, we have the following:*

$$\mathsf{Pr}[\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, f, \mathsf{ct}_1, \ldots \mathsf{ct}_\ell)) \neq f(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)] \leq \mathrm{negl}(\lambda).$$

Compactness: *There exists a polynomial* $p(\lambda)$ *such that, for any (Boolean) function* $f : \{0,1\}^\ell \to \{0,1\} \in \mathcal{F}$ *and any sequence of* ℓ *messages* $\mathsf{m}_1, \ldots, \mathsf{m}_\ell$*, letting* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*, and* $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_i)$ *for each* $i \in [\ell]$*, we have*

$$|\mathsf{ct}^* \leftarrow \mathsf{Eval}(\mathsf{pk}, f, \mathsf{ct}_1, \ldots \mathsf{ct}_\ell))| \leq p(\lambda),$$

where $p(\lambda)$ *is independent of size of* f *and the number* ℓ *of inputs.*

In the definition mentioned above, we assumed that the evaluation key is included as part of the public key.

3.3 Threshold FHE

In this section, we define Threshold FHE (or ThFHE in short), which is aligned with ThFHE definition of [BGG+18].

Definition 2 (Threshold Fully Homomorphic Encryption (ThFHE)). *Let* $\mathbb{S}$ *be a class of efficient access structures on a set of parties* $\mathcal{P} = \{P_1, \ldots, P_T\}$*. A* ThFHE *scheme for* $\mathbb{S}$ *over a message space* $\mathcal{M}$ *is a tuple of probabilistic polynomial-time algorithms*

$$\mathsf{ThFHE} = (\mathsf{ThFHE.Gen}, \mathsf{ThFHE.Enc}, \mathsf{ThFHE.Eval}, \mathsf{ThFHE.PartialDec}, \mathsf{ThFHE.Combine}),$$

defined as follows:

- $\mathsf{ThFHE.Gen}(1^\lambda, 1^d, \mathbb{A})$*: On input the security parameter* λ*, a depth bound* d*, and an access structure* $\mathbb{A} \in \mathbb{S}$*, the setup algorithm outputs an encryption (public) key* pk*, a decryption (secret) key* sk*, and a set of secret key shares* $\mathsf{sk}_1, \ldots, \mathsf{sk}_T$*.*
- $\mathsf{ThFHE.Enc}(\mathsf{pk}, \mu)$*: On input* pk *and a plaintext* μ*, the encryption algorithm outputs a ciphertext* ct*.*

- $\mathsf{ThFHE.Eval}(\mathsf{pk}, \mathsf{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$: *On input a public key* pk*, a (Boolean) circuit* C *of depth at most* d[1]*, and a set of ciphertexts* $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$*, the evaluation algorithm outputs a ciphertext* ct^**.*

- $\mathsf{ThFHE.PartialDec}(\mathsf{sk}_i, \mathsf{ct})$: *On input a secret key share* sk_i *and a ciphertext* ct*, the partial decryption algorithm outputs a partial decryption* p_i*.*

- $\mathsf{ThFHE.Combine}(\{p_i\}_{i\in\mathcal{S}})$: *On input a set of partial decryptions* $\{p_i\}_{i\in\mathcal{S}}$ *for some subset* $\mathcal{S} \subseteq \{P_1, \ldots, P_T\}$*, the combination algorithm either outputs a plaintext* μ *or the symbol* $\perp$*.*

Correctness and Compactness. We require that a ThFHE scheme satisfies compactness and correctness. These properties are very similar in flavor to those for traditional FHE 3.2. In case of correctness, while FHE requires that any honestly generated ciphertext should be decrypted to the correct plaintext, ThFHE requires that given an honestly generated ciphertext from homomorphic evaluation of some circuit on some encrypted inputs, recombining its partial decryptions by a threshold number of parties should result in correct circuit output. See [BGG$^+$18] for formal definitions.

IND-secure ThFHE [JRS17]. In this paper, we adopt the definition of IND-secure ThFHE from [JRS17]. We subsequently present a discussion on this notion of security.

Consider a ThFHE scheme over a message space $\mathcal{M}$ for a threshold access structure $\mathbb{A}_{t,T}$ for a set of T parties $\mathcal{P} = \{P_1, \ldots, P_T\}$. Let λ be the security parameter and d be the depth bound for the ThFHE scheme. We define below a game $\mathsf{G}_{\mathsf{ThFHE},\mathcal{A},\mathbb{A}_{t,T}}(1^\lambda, 1^d)$ between a PPT challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$.

$\underline{\mathsf{G}_{\mathsf{ThFHE},\mathcal{A},\mathbb{A}_{t,T}}(1^\lambda, 1^d)}$:

Initialization phase.

1. The challenger $\mathcal{C}$ runs $\mathsf{ThFHE.Gen}(1^\lambda, 1^d, \mathbb{A}_{t,T})$ to obtain $(\mathsf{pk}, \mathsf{sk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_T)$, such that sk_i is the secret share corresponding to P_i, and provides the public key pk to the adversary $\mathcal{A}$.

2. The adversary $\mathcal{A}$ outputs a set $\mathcal{S} \subset \{P_1, \ldots, P_T\}$ such that $\mathcal{S} \notin \mathbb{A}_{t,T}$, and receives the set of secret key shares $\{sk_i\}_{P_i\in\mathcal{S}}$ from $\mathcal{C}$.

Challenge phase.

1. The adversary $\mathcal{A}$ outputs two set of messages $\mathbf{m}_0 = (m_1^0, \ldots, m_\ell^0), \mathbf{m}_1 = (m_1^1, \ldots, m_\ell^1) \in \mathcal{M}$.

2. The challenger $\mathcal{C}$ randomly samples a bit $b \leftarrow \{0, 1\}$ and provides $\mathcal{A}$ with $\mathbf{ct}^\star = \{\mathsf{ct}_i^\star\}_{i\in[\ell]} = \{\mathsf{ThFHE.Enc}(\mathsf{pk}, m_i^b)\}_{i\in[\ell]}$.

Partial decryption query phase.

[1]As we deal with FHE with bootstrapping in Torus-FHE, any circuit with arbitrary depth can be evaluated.

2. The adversary $\mathcal{A}$ issues Q ($= \text{poly}(\lambda)$) circuits $\{C_i : \mathcal{M}^\ell \to \mathcal{M}\}_{i\in[Q]}$, such that $C_i(m_1^0, \ldots, m_\ell^0) = C_i(m_1^1, \ldots, m_\ell^1)\ \forall i \in [Q]$.

2. In response, $\mathcal{A}$ receives $\left(\hat{\mathsf{ct}}_i, \{p_{i,j}\}\right)$ for each $i \in [Q]$ and $P_j \notin \mathcal{S}$, where

$$\hat{\mathsf{ct}}_i = \mathsf{ThFHE.Eval}(\mathsf{pk}, C_i, \{\mathsf{ct}_j^\star\}_{j\in[\ell]}), \qquad p_{i,j} = \mathsf{ThFHE.PartialDec}(\mathsf{sk}_j, \hat{\mathsf{ct}}_i).$$

Output phase.

1. The adversary $\mathcal{A}$ eventually outputs a bit $b' \in \{0,1\}$.

2. if $b' = b$, the game outputs 1, otherwise it outputs 0.

Note that during the "partial decryption query phase" we do not allow evaluation of such circuits, whose output differs upon whether its input is $\mathbf{m}_0$ or $\mathbf{m}_1$; otherwise, the adversary can trivially win by observing the decrypted output of such circuits. Note that the adversary would know the (threshold)-decrypted value of evaluated ciphertext, as it is able to compute the partial decryptions of the corrupted parties on its own and it can get the partial decryptions of the honest parties from partial decryption query, and then combine them to get the final decryption.

We say that a threshold FHE scheme ThFHE is IND-secure if, for any security parameter $\lambda \in \mathbb{N}$, for any depth $d = \text{poly}(\lambda)$, for any threshold access structure $\mathbb{A}_{t,T}$, and for any PPT adversary $\mathcal{A}$, letting $\gamma_\beta = \Pr[\mathsf{G}_{\mathsf{ThFHE},\mathcal{A},\mathbb{A}_{t,T}}(1^\lambda, 1^d) = \beta]$, for $\beta \in \{0,1\}$ (where the probability is over the random coins used by $\mathsf{ThFHE.Gen}$, $\mathsf{ThFHE.Enc}$, $\mathsf{ThFHE.Eval}$ and the adversary $\mathcal{A}$), we have $\left|\gamma_0 - \gamma_1\right| \leq \text{negl}(\lambda)$.

Discussion on IND-Security of ThFHE. The IND-security definition of ThFHE from [JRS17] effectively combines in sequence the definitions of simulation and semantic security for ThFHE from prior works [MW16, BGG$^+$18, CCK23]. Informally, a ThFHE scheme is said to provide semantic security if a PPT adversary cannot efficiently distinguish between encryptions of arbitrarily chosen plaintext messages $\mathbf{m}_0$ and $\mathbf{m}_1$ [BGG$^+$18, CCK23]. Additionally, a ThFHE scheme is said to provide simulation security if there exists an efficient algorithm to *simulate* partial decryptions of honest parties on ciphertexts that are produced by evaluating one or more circuits on (honestly generated) ciphertexts, without any knowledge of secret shares of honest parties [MW16, BGG$^+$18, CCK23]. The IND-security definition of [JRS17] can be viewed as an indistinguishability-based security notion where we essentially require these two notions of security to hold *simultaneously* against a PPT adversary that is given the secret key shares of a set of corrupt parties belonging to an *invalid access structure set* $\mathcal{S}$.

As noted in [JRS17], IND-security is a natural notion of security for ThFHE, and is implied by the simulation security definitions in prior works [MW16, BGG$^+$18]. As is the case for many other cryptographic primitives (e.g., functional encryption [BSW11]), indistinguishability-based security suffices for real-world applications (e.g., it suffices for the application presented subsequently as part of our case study in Section 6). Hence, we adopt this definition from [JRS17] for our work, and prove the security of our proposed ThFHE scheme under this definition.

Notably, the IND-security definition allows us to port existing technical machinery for Rényi divergence-based analysis of other lattice-based cryptosystems [BLRL$^+$18] to the context of threshold FHE (unfortunately, the original simulation security definition from prior works [MW16, BGG+18, CCK23] is not amenable to such techniques). Rényi divergence has previously been applied to achieve better parameter choices, particularly in case of search problems, for e.g., [ASY22, BLRL+18, BGM+16]. Applying Rényi divergence in the context of distinguishing problems is not straightforward. However, in this work, we can use techniques from [BLRL+18] to argue that for any ThFHE scheme, as long as the adversary's views of the real and simulated partial decryptions in our security game are publicly sampleable and have a bounded Rényi divergence, it cannot distinguish between encryptions of $\mathbf{m}_0$ and $\mathbf{m}_1$ with non-negligible probability without breaking the original semantic security guarantees of underlying FHE scheme. Looking ahead, for our proposed ThFHE scheme, we can achieve the desired bounds on the Rényi divergence while only using a polynomial modulus-to-noise ratio, which is the technical crux of our contribution. On the contrary, it is seemingly hard to achieve the original notion of simulation security proposed in [MW16, BGG+18, CCK23] without a superpolynomial modulus-to-noise ratio.

Relation with IND-CPA^D Security of Approximate FHE. A (seemingly) related notion of IND-CPA^D security emerged in order to make approximate homomorphic encryption schemes secure against a specific key-recovery attack [LM21], which exploits the fact that a decryption oracle access to the adversary for the honestly generated ciphertexts helps it to retrieve the ciphertext noise in this scenario. However, the notion of IND-CPA^D security reduces to IND-CPA security for the exact homomorphic schemes [LM21]. And as we deal with exact fully homomorphic encryption schemes (albeit in the threshold setting), we do not provide exactly that decryption oracle access (for honestly generated ciphertexts) to the adversary. Instead, we allow the adversary to query for honest parties' partial decryption on honestly generated ciphertexts with proper constraints as described in the game $\mathsf{G}_{\mathsf{ThFHE},\mathcal{A},\mathbb{A}_{t,T}}(1^\lambda, 1^d)$. Therefore, though both definitions are augmented from standard IND-CPA security, there are crucial differences in the settings that seemingly make them orthogonal. For IND-CPA^D, the entire secret key is in one place and the decryption oracle performs the entire decryption and then returns an erroneous plaintext, whereas, in the IND-secure security definition, a partial decryption oracle just returns a (possibly noisy) partial ciphertext computed using a share of the key. One may notice that a special case of IND-secure security, where there is only one party (essentially a centralized FHE) coincides with the standard IND-CPA, as we are in the exact setting. However, as long as the secret key is shared between more than one party, IND-secure security appears to become orthogonal to IND-CPA^D, despite high-level similarities in terms of allowing decryption.[1]

3.4 Linear Integer Secret Sharing Scheme (LISSS)

In this work, we base our constructions and software implementation of Threshold FHE on a special class of secret sharing schemes called Linear Integer Secret Sharing Scheme (LISSS) defined below.

Definition 3 (LISSS)**.** *Let $\mathcal{P} = \{P_1, \ldots, P_T\}$ be a set of parties, and let $\mathbb{S}$ be a class of efficient access structures on $\mathcal{P}$. A secret sharing scheme* SS *with secret space $\mathcal{K} = \mathbb{Z}_p$ for some prime p is called a linear integer secret sharing scheme (LISSS) if there exist the following algorithms:*

[1] Though we do not provide formal proof of orthogonality, one can observe that, as long as the secret key is shared, it is not clear how a reduction can work.

- $\mathsf{SS.Share}(k \in \mathcal{K}, \mathbb{A})$: *There exists a matrix* $\mathbf{M} \in \mathbb{Z}_p^{d \times e}$ *with dimensions determined by the access structure* $\mathbb{A} \in \mathbb{S}$ *called the distribution matrix, and each party* P_i *is associated with a partition* $T_i \subseteq [d]$. *To create the shares on a secret* $k \in \mathcal{K}$, *the sharing algorithm uniformly samples* $\rho_2, \ldots, \rho_e \leftarrow \mathbb{Z}_p$, *defines a vector* $\mathbf{s} = (s_1, \ldots, s_d)^{\mathbf{T}} = \mathbf{M} \cdot (k, \rho_2, \ldots, \rho_e)^{\mathbf{T}}$, *and outputs to each party* P_i *the corresponding set of shares* $\mathsf{share}_i = \{s_j\}_{j \in T_i}$.
- $\mathsf{SS.Combine}(\{\mathsf{share}_i\}_{P_i \in \overline{\mathcal{P}}})$: *For any qualified subset of parties* $\overline{\mathcal{P}} \in \mathbb{A}$, *there exists a set of efficiently computable "recovery coefficients"* $\{c_j\}_{j \in \cup_{P_i \in \overline{\mathcal{P}}} T_i}$, *such that*

  $$\sum_{j \in \cup_{P_i \in \overline{\mathcal{P}}} T_i} c_j \cdot \mathbf{M}[j] = (1, 0, \ldots, 0),$$

  where $\mathbf{M}[j]$ *denotes the* j*-th row of the matrix* $\mathbf{M}$ *described earlier. Then, the final secret* k *can be re-computed using these recovery coefficients as*

  $$k = \sum_{j \in \cup_{P_i \in \overline{\mathcal{P}}} T_i} c_j \cdot s_j.$$

Definition 4 ($\{-1,0,1\}$-LISSS)**.** *Let* $\mathcal{P} = \{P_1, \ldots, P_T\}$ *be a set of parties, and let* $\mathbb{S}$ *be a class of efficient access structures on* $\mathcal{P}$. *Any LISSS scheme* $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Combine})$ *as defined above is a* $\{-1,0,1\}$*-LISSS if it is guaranteed that for any set of "recovery coefficients"* $\{c_j\}$ *generated by* $\mathsf{SS.Combine}$ *(on input the set of shares corresponding to a qualified subset of parties* $\overline{\mathcal{P}} \in \mathbb{A}$ *for an access structure* $\mathbb{A} \in \mathbb{S}$*), we must have* $c_j \in \{-1,0,1\}$.

In this paper, we use a special instance of $\{-1,0,1\}$-LISSS, called the Benaloh-Leichter LISSS [DT06]. We expand more on Benaloh-Leichter LISSS in Section 4.3.

4 Our Proposal: Torus-FHE with Threshold Decryption

In this section, we present our construction of first practical threshold FHE. We introduce two protocols - threshold secret sharing of the decryption key and threshold decryption, to realize our final ThFHE. Along the way, we describe our two main theoretical contributions - an extension of the standard LISSS secret sharing scheme due to Benaloh and Leichter [DT06] to support the secret key structure which consists of binary polynomials, and the usage of Rényi Divergence based analysis to achieve only a small polynomial blowup in the noise level for our proposed threshold ThFHE built upon Torus-FHE scheme. We first describe the generic decryption algorithm of any Ring-LWE based FHE scheme and then build its thresholdized construction. Our security proofs rely on the hardness of the LWE problem in the ring setting with binary secrets.

Remark. We remark here that our threshold decryption technique can, in fact, be generalized to any lattice-based encryption scheme where the decryption procedure involves computing a linear function of the secret key (in particular, Regev-style decryption based on computing an inner-product of the ciphertext vector and the secret key vector). However, since our concrete goal is to realize a threshold version of the Torus-FHE scheme from [CGGI20], we keep our theoretical discussion aligned with the Torus-FHE scheme (and the Torus-FHE library) for ease of exposition.

4.1 Decryption in Torus-FHE

For ease of exposition, we start with describing the generic decryption algorithm of a Ring-LWE based Torus-FHE scheme over a message space $\mathcal{M} = \mathbb{T}[X]/(X^N + 1)$. We assume TRLWE to be an instantiation of such a scheme, represented by a tuple of PPT algorithms as follows,

$$\mathsf{TRLWE} = (\mathsf{TRLWE.Gen}, \mathsf{TRLWE.Enc}, \mathsf{TRLWE.Eval}, \mathsf{TRLWE.Dec}).$$

The scheme has two fixed parameters N and k to denote size of polynomials and number of polynomials respectively. The secret key (say, $\mathbf{SK}$) in TRLWE has the following structure with $SK_{i,j} \in \{0,1\}\ \forall 1 \le i \le k, \forall 1 \le j \le N$,

$$\mathbf{SK} = \left(\sum_{j=1}^{N} SK_{1,j}x^{j-1}, \ldots, \sum_{j=1}^{N} SK_{k,j}x^{j-1}\right).$$

The ciphertext in TRLWE can be written as $\mathsf{CT} = (A, B)$, where $B = \sum_{i=1}^{k} A[i]\cdot\mathbf{SK}[i]+\mathsf{m}+e$. Here m is the underlying plaintext and A can be represented as the following with each $A_{i,j} \in \mathbb{T}$,

$$A = \left(\sum_{j=1}^{N} A_{1,j}x^{j-1}, \ldots, \sum_{j=1}^{N} A_{k,j}x^{j-1}\right).$$

Also, $A[i] \cdot \mathbf{SK}[i]$ is the polynomial multiplication between i^{th} polynomial of A and i^{th} polynomial of $\mathbf{SK}$ modulo $(x^N + 1)$. In order to avoid notational complexity, we will henceforth use $A \cdot \mathbf{SK}$ to denote $\sum_{i=1}^{k} A[i] \cdot \mathbf{SK}[i]$ in the paper. And, $e = \sum_{j=1}^{N} e_j x^{j-1}$ is RLWE noise polynomial with each $e_j \leftarrow \mathcal{G}$, where $\mathcal{G}$ is a Gaussian distribution.

We focus on distributed decryption of a Ring-LWE ciphertext and rely on a public key adaptation [Rot11] of underlying FHE scheme to perform the encryption and evaluation operations. Hence we do not discuss those algorithms (TRLWE.Enc, TRLWE.Eval) here. We discuss the original decryption algorithm TRLWE.Dec first, we modify it later in order to support threshold decryption.

$\mathsf{TRLWE.Dec}(\mathbf{SK}, \mathsf{CT})$: Given the secret key $\mathbf{SK}$ and a ciphertext $\mathsf{CT} = (A, B)$, the decryption algorithm proceeds in two steps as follows:

- $\mathsf{TRLWE.Decode}_0(\mathbf{SK}, \mathsf{CT})$: On input ciphertext CT and secret key $\mathbf{SK}$, this step of the decryption calculates $\Phi = B - A \cdot \mathbf{SK}$, which is equal to $\mathsf{m} + e$. Here, m is the underlying plaintext and e is a Torus ring-LWE noise polynomial.
- $\mathsf{TRLWE.Decode}_1(\Phi)$: This final step rounds up each of the N coefficients of Φ to return the "exact" coefficients of the plaintext message m.

The security of TRLWE follows from the hardness of the Binary Ring Learning with Errors (BRLWE) problem (see Section 3 for the formal definition). Note that although a reduction from binary LWE to LWE exists [Mic18], a reduction for its ring-variant is not yet known; nonetheless, binary RLWE is widely [BBPS19, BD20] believed to be computationally hard.

Our main contribution is a proposal for thresholdizing the decryption of the aforementioned TRLWE scheme. We discuss the specific case of (T, T)-threshold decryption and its security analysis based on Rényi Divergence in subsequent sections. We provide the generalized (t, T)-threshold decryption construction in Section 4.4, and related the security analysis in Section 4.6.

4.2 Achieving (T, T)-Distributed Decryption

Let us assume $\mathcal{P} = \{P_1, \ldots, P_T\}$ is the set of T parties and they are willing to perform $\mathsf{TRLWE.Dec}$ on a Torus Ring-LWE ciphertext $\mathsf{CT} = (A, B)$ in a distributed way. We are in the *dealer-based model*, i.e., we assume that a trusted dealer uses some secret sharing algorithm to distribute the Torus Ring-LWE secret $\mathbf{SK}$ to each P_i as SH_i, such that $\mathbf{SK} = \sum_{i=1}^{T} SH_i$. In this context, each $P_i \in \mathcal{P}$ individually performs the following steps:

- $\mathsf{TRLWE.PartialDec}(SH_i, \mathsf{CT})$: On input of the secret share SH_i and the ciphertext $\mathsf{CT} = (A, B)$, this algorithm generates partially decrypted ciphertext $part\_decrypt_i = A \cdot SH_i + e^i_{sm}$. Here, e^i_{sm} is the smudging noise polynomial added by P_i, where each coefficient of e^i_{sm} is sampled from the Gaussian smudging noise distribution $\mathcal{G}_{sm}$ (we expand on the smudging noise subsequently in Section 4.5). The partial decryption $part\_decrypt_i$ is then broadcast to the rest of the $(T-1)$ parties.

- $\mathsf{TRLWE.Combine}(\{part\_decrypt_i\}_{i\in[T]}, \mathsf{CT})$: This algorithm takes as input of all the partially decrypted ciphertexts $part\_decrypt_i$ (where $i \in [T]$) and the ciphertext $\mathsf{CT} = (A, B)$, and combines them as $\Phi = B - \sum_{i=1}^{T} part\_decrypt_i$. Note that Φ is essentially $\left(\mathsf{m} + e - \sum_{i=1}^{T} e^i_{sm}\right)$.

- $\mathsf{TRLWE.Decode}_1(\Phi)$: On input of the phase Φ, each of its N coefficients are rounded up to retrieve N coefficients of the message m.

Clearly, this (T, T) distributed decryption is very specific, as participation of each party is mandatory to perform a distributed decryption. Next, we generalize this to (t, T) threshold decryption for any $0 < t < T$. Our proposal relies on a (t, T) threshold secret sharing, which is an extended version of the original Benaloh-Leichter LISSS [DT06] and is elaborated in Section 4.3.

4.3 Extending Benaloh-Leichter LISSS

We next aim to generalize the aforementioned threshold decryption protocol to support (t, T)-threshold decryption for any $t \leq T$, which requires an appropriate LISSS (see Section 3.4) to support (t, T)-threshold secret sharing. For this purpose, we resort to using Benaloh-Leichter LISSS [DT06], as it supports efficient final combination of partial decryptions, in contrast to multiplying partial decryptions with large Lagrange coefficients while using Shamir's secret sharing [Sha79], that leads to noise-blowup in the ciphertext and incorrect decryption. Original scheme shares a scalar secret, but the secret of TRLWE is composed of k number of N-sized binary polynomials as described in Section 4.1. Hence, we propose an extended Benaloh-Leichter (t, T)-threshold secret sharing scheme to support Torus-RLWE secret key sharing. Let $\mathbf{SK}$ be the Torus-RLWE secret key, which is to be shared among T parties belonging to the set $\mathcal{P} = \{P_1, \ldots, P_T\}$. We first describe some pre-processing steps required for (t, T)-threshold secret sharing.

Formation of Distribution Matrix M. Formation of distribution matrix M depends upon the monotone Boolean formula (MBF[1]), representing a (t, T)-threshold access structure. Also, any MBF, being a combination of AND and OR of Boolean variables, we are able to construct distribution matrix of any monotone Boolean formula by taking care of the following three cases:
A Boolean variable. I_k, the identity matrix of dimension k, represents the distribution matrix of each Boolean variable x_i.

AND-ing of two MBFs. Let us suppose, matrix M_{f_a} and M_{f_b} are the distribution matrices for MBFs f_a and f_b respectively and have dimension $d_a \times e_a$ and $d_b \times e_b$ respectively. Then we form $M_{f_a \wedge f_b}$ to represent $f_a \wedge f_b$ as follows:

c_a^k	c_a^k	C_a	0
0	c_b^k	0	C_b

Here, c_a^k and c_b^k denote first k columns and C_a and C_b denote the rest of the columns of M_{f_a} and M_{f_b} respectively. Resulting $M_{f_a \wedge f_b}$ has dimension $(d_a + d_b) \times (e_a + e_b)$.

OR-ing of two MBFs. Assuming matrices M_{f_a} and M_{f_b} of dimension $d_a \times e_a$ and $d_b \times e_b$ respectively to be the distribution matrices for Boolean formula f_a and f_b respectively. Then we form $M_{f_a \vee f_b}$ of dimension $(d_a + d_b) \times (e_a + e_b - k)$ to represent $f_a \vee f_b$ as following:

c_a^k	C_a	0
c_b^k	0	C_b

Here, c_a^k and c_b^k denote first k columns of M_{f_a} and M_{f_b} respectively. C_a and C_b denote the rest of the columns of M_{f_a} and M_{f_b} respectively.
It can be easily verified that, the distribution matrix M for (t, T)-threshold secret sharing has dimension $d \times e$, where $d = \binom{T}{t}kt$ and $e = (\binom{T}{t}kt - (\binom{T}{t} - 1)k)$.

Formation of Share Matrix $\boldsymbol{\rho}$. Though ρ is a vector in the original scheme [DT06], in our extended version, ρ is a matrix with dimension $e \times N$. Its first k rows are populated from the coefficients of k binary polynomials in **SK**. The rest of the rows of the matrix are filled uniformly randomly from $\{0, 1\}$.

Sharing. We provide the secret sharing algorithm in Algorithm 1. The number of t-sized subsets of $\mathcal{P}$ is $\binom{T}{t}$. We enumerate over all these subsets and tag each of them with corresponding enumerating serial number and call it the group_id. Once the sharing process is complete, each party P_i gets $\binom{T-1}{t-1}$ number of key shares to store, for each possible t-sized group, that P_i can belong to. To differentiate among these key shares, we tag each key share with following two attributes:

- party_id: refers to which party the key share belongs to.
- group_id: refers to the t-sized group for the key share.

[1]By MBF, we refer to Boolean formulae having a single output and consisting of only AND and OR combination of variables.

Algorithm 1 t-out-of-T Secret Sharing

```
function ShareSecret(t, T, M, ρ, d, k)
    shares ← M · ρ
    row ← 1
    while row ≤ d do
        gid ← ⌈row/kt⌉
        pt ← findParties(gid, t, T)
        for i = 1 to t do
            rowcount ← row + (i − 1)k
            curr_share ← TRLweKey()                ▷ New TRLWE Key
            for j = 0 to k − 1 do
                curr_share[j] ← shares[rowcount + j]
            cur_share.party_id ← pt[i − 1]
            cur_share.group_id ← gid
        row ← row + kt
```

Total $d = \binom{T}{t}kt$ rows of *shares* matrix, produces $\binom{T}{t}t$ number of key shares, each tagged with specific group_id and party_id. The findParties(gid, t, T) procedure in Algorithm 1, returns a list of party ids present in gid^{th} t-sized group(subset) of $\mathcal{P}$.

Reconstruction. Any t-sized group of parties should be able to reconstruct **SK**, with the help of the key shares, they have. Given a t-sized group $\mathcal{P}' = \{P'_1, P'_2, \ldots, P'_t\} \subset \mathcal{P}$, each of the t parties will have one key share with group_id corresponding to $\mathcal{P}'$. Let us denote these t key shares as $\{SH_1, SH_2, \ldots, SH_t\}$. We observe (Appendix A) that exactly one share among them will have non-binary coefficients in its k polynomials. We call the party, having non-binary key share, the group_leader of the t-sized group. In any t-sized group, the party with minimum value of party_id is the group_leader.
Now, without loss of generality, let us assume P'_1 is the group_leader of $\mathcal{P}'$ and its non-binary key share is SH_1. Then the secret S can be reconstructed as: $\mathbf{SK} = SH_1 - \sum_{i=2}^{t} SH_i$. Hence, recovery coefficient c_1 is 1 for the group_leader and c_i is -1 for each of other $(t-1)$ parties. We exploit this reconstruction property in final combination stage of (t, T)-threshold decryption technique.

Size of Secret Shares. After applying (t, T)-threshold secret sharing on **SK**, each party gets $\binom{T-1}{t-1}$ key shares to store. For any t-sized group, the group_leader's share size (in number of bits) is upper bounded by $\lceil \log_2 t \rceil \cdot N \cdot k$, and each of the other $(t-1)$ parties has share of size exactly $N \cdot k$ bits. This can be proved by close observation of the secret shares (See Appendix A).

4.4 Generalized (t, T)-Threshold Decryption Protocol

In this section, we describe the generalized (t, T)-threshold decryption algorithm for our proposed threshold Torus-FHE. Here, we use the extended version of Benaloh-Leichter LISSS, proposed in Section 4.3 to share the secret key across the various parties (as opposed to a simple additive sharing in the (T, T)-case in Section 4.2). Consequently, we need to modify the TRLWE.PartialDec and TRLWE.Combine algorithms to enable correct and efficient decryption by any t'-sized subset of the T parties for $t' \geq t$.

Let $\mathcal{P} = \{P_1, \ldots, P_T\}$ be any set of T parties and let $\mathcal{P}' = \{P_{id_1}, \ldots, P_{id_t}\} \subset \mathcal{P}$ be a t-sized subset of $\mathcal{P}$ with $\mathsf{group\_id}$ j, authorized to threshold-decrypt a ciphertext $\mathsf{CT} = (A, B)$. Also, without loss of generality, let us assume $id_1 < \cdots < id_t$, so that P_{id_1} is the $\mathsf{group\_leader}$ of $\mathcal{P}'$.

We begin by assuming that all of the T parties in $\mathcal{P}$ have already received their key shares after successful execution of the (t, T)-threshold secret sharing scheme on $\mathbf{SK}$ (Section 4.3). Hence, each $P_{id_i} \in \mathcal{P}'$ has exactly one key share corresponding to $\mathsf{group\_id}$ j. We denote these t key shares as $\{SH_{id_1,j}, \ldots, SH_{id_t,j}\}$. Let us recall from Section 4.3 that,

$$\mathbf{SK} = SH_{id_1,j} - \sum_{i=2}^{t} SH_{id_i,j}.$$

The threshold decryption of CT consists of the following steps, performed by each $P_{id_i} \in \mathcal{P}'$ individually:

- $\mathsf{TRLWE.PartialDec}(SH_{id_i,j}, \mathsf{CT})$: On input Torus Ring-LWE ciphertext CT and a key share $SH_{id_i,j}$, P_{id_i} calculates the following:

  $$part\_decrypt_{id_i} = A \cdot SH_{id_i,j} + e_{sm}^{id_i},$$

  where $e_{sm}^{id_i}$ is a smudging noise polynomial and each coefficient of $e_{sm}^{id_i}$ is sampled from a Gaussian smudging noise distribution $\mathcal{G}_{sm}$. Then, P_{id_i} broadcast $part\_decrypt_{id_i}$ to rest of the $(t-1)$ parties.

- $\mathsf{TRLWE.Combine}(\{part\_decrypt_{id_i}\}_{i \in [t]}, \mathsf{CT})$: On input all t partial decryptions, each party calculates the phase

  $$\phi = B - (part\_decrypt_{id_1} - \sum_{i=2}^{t} part\_decrypt_{id_i}),$$

  where ϕ equals $\mathsf{m} + e - e_{sm}^{id_1} + \sum_{i=2}^{t} e_{sm}^{id_i}$.

- $\mathsf{TRLWE.Decode}_1(\phi)$: Each of the N coefficients of ϕ is rounded up to extract the coefficients of the message m.

Properties. The correctness and compactness of the proposed (t, T)-threshold decryption scheme directly follows from the proofs of threshold-FHE scheme mentioned in Section 3.3. Its proof of security is presented in Section 4.6.

4.5 Polynomial Modulus-to-Noise Ratio via Rényi Divergence

We now elaborate on our main theoretical contribution, namely, achieving a polynomial modulus-to-noise ratio (i.e. a polynomial ratio between the modulus q and the Ring LWE noise e) for our proposed threshold version of Torus-FHE (abbreviated as TRLWE henceforth) via: (a) a novel usage of Gaussian smudging noise during partial decryption (as described earlier in Section 4.2 and Section 4.4), and (b) application of Rényi Divergence for distinguishing problems with public sampleability property to prove the security of our proposed

Threshold FHE scheme TRLWE (under our proposed security definition in Section 3.3) as well as to get efficient choice of parameters for the scheme.

Our Approach: Rényi Divergence-based Analysis of Smudging Noise. In this paper, due to our novel approach of using Gaussian smudging noise and then using a Rényi Divergence based analysis akin to that of [BLRL+18, TT15] as opposed to the statistical distance based analysis used in prior works [MW16, BGG+18, CCK23], it suffices to sample the smudging noise from a Gaussian distribution with standard deviation only polynomially larger than the standard deviation of the Gaussian distribution pertaining to the RLWE noise. As a result, from a theoretical point of view, we obtain the *first practical* ThFHE *scheme with polynomial modulus to noise ratio*. From an implementation point of view, it leads to a massive improvement in the practical performance of our prototype implementation in software (presented in Section 5). We expand on our approach below.

Analyzing (T, T)-Distributed Decryption. For the ease of exposition, we now describe the Rényi Divergence-based analysis of our proposed distributed decryption protocol for TRLWE for the special case of (T, T)-distributed decryption (described originally in Section 4.2). We defer the analysis of the more general case of (t, T)-threshold decryption to Section 4.6.

The Adversarial Model. Recall from Section 4.2 that for the case of (T, T)-distributed decryption, the Torus Ring-LWE secret $\mathbf{SK}$ is linearly secret-shared across $\{P_1, \ldots, P_T\}$ as $\mathbf{SK} = \sum_{i=1}^{T} SH_i$, where party P_i holds the secret key share SH_i. Now consider a scenario where, as per our security definition in Section 3.3, an adversary $\mathcal{A}$ corrupts all but one party (say party P_1 without loss of generality), and gains access to the secret key shares of all of the corrupted parties (i.e., $SH_2, \ldots, SH_T$). Keeping analogy to our security definition, assume that $\mathcal{A}$ chooses two set of plaintexts $\mathbf{M}_0 = \{M_i^0\}_{i\in[\ell]}$ and $\mathbf{M}_1 = \{M_i^1\}_{i\in[\ell]}$ and receives a challenge set of honest encryptions $\mathbf{CT}^\star = \{\mathsf{CT}_i^\star\}$, which is component-wise encryption of either $\mathbf{M}_0$ or $\mathbf{M}_1$.

Now suppose that $\mathcal{A}$ issues Q-many partial decryption queries. In each query, it provides a new circuit C of bounded depth to a challenger with a constraint that $C(\{M_i^0\}_{i\in[\ell]}) = C(\{M_i^1\}_{i\in[\ell]})$. The challenger computes a resultant ciphertext $\widehat{\mathsf{CT}} = (A, B)$ by homomorphically evaluating C on the set $\mathbf{CT}^\star$. The adversary $\mathcal{A}$ is then allowed to see the partial decryption of $\widehat{\mathsf{CT}}$ by the honest party P_1, computed (in the "real" security game) as $part\_decrypt_1 = A \cdot SH_1 + e_{sm}$, where e_{sm} is the smudging noise polynomial added by party P_1 (each coefficient is sampled from a Gaussian distribution $\mathcal{G}_{sm}$ with standard deviation σ).

"Simulating" an Honest Partial Decryption. We now construct a simulator $\mathcal{S}$ that "simulates" a partial decryption of $\widehat{\mathsf{CT}}$ on behalf of the honest party P_1 *without* the knowledge of the partial decryption key SH_1, but simply from the knowledge of the underlying plaintext m and the knowledge of the corrupted partial decryption keys $\{SH_j\}_{j\in[2,T]}$. Before delving into the description of the simulator $\mathcal{S}$, we briefly motivate the construction of such a simulator $\mathcal{S}$. Observe that $\mathcal{S}$ has no additional information beyond what $\mathcal{A}$ already knows. So, $\mathcal{A}$ is not able to distinguish $\mathbf{CT}_0$ from $\mathbf{CT}_1$, i.e., the component-wise encryption of two set of plaintexts $\mathbf{M}_0$ and $\mathbf{M}_1$ of its choice, due to the hardness of Binary Ring-LWE assumption, on which the original Torus-FHE scheme relies.

We now construct the simulator $\mathcal{S}$ as follows. Given the ciphertext $\widehat{\mathsf{CT}} = (A, B)$, its underlying plaintext message m, and the corrupted partial decryption keys $\{SH_j\}_{j\in[2,T]}$, the simulator $\mathcal{S}$ outputs a "simulated" partial decryption

$$part\_decrypt_1^{\mathsf{Sim}} = B - \mathsf{m} - \sum_{i=2}^{T} A \cdot SH_i + e_{sm},$$

where e_{sm} is a smudging noise polynomial (again, each coefficient of this polynomial is sampled from a Gaussian distribution $\mathcal{G}_{sm}$ with standard deviation σ). Now, observe that, letting $\gamma = B - \mathsf{m} - \sum_{i=2}^{T} A \cdot SH_i$, we have

$$part\_decrypt_1 = \gamma - e + e_{sm}, \qquad part\_decrypt_1^{\mathsf{Sim}} = \gamma + e_{sm},$$

where e is the RLWE noise polynomial embedded in $\widehat{\mathsf{CT}}$.

Rényi Divergence-based Analysis. Let η be the set of fixed parameters instantiating the security game described in Section 3.3 as follows.

$$\eta = (\mathbf{PK}, \mathbf{SK}, \{\mathbf{SK}_i\}_{i\in[T]}, \mathbf{M}_0, \mathbf{M}_1, \{C_i\}_{i\in[Q]}).$$

Let a distribution

$$D_b^{\eta}(r) = (\mathbf{PK}, \{\mathbf{SK}_i\}_{i\in[2,T]}, \mathbf{M}_0, \mathbf{M}_1, \mathbf{CT}_b, \{\widehat{\mathsf{CT}}_i^{b}\}_{i\in[Q]}, \{p_i^b\}_{i\in[Q]})$$

represent the view of the adversary, when the challenger $\mathcal{C}$ samples bit b in the challenge phase. The set of noise values $r = \{r_i\}_{i\in[Q]}$ is used in the computation of honest party P_1's partial decryption $p_i = \gamma + r_i$ and each r_i is sampled either from distribution of $(e_{sm} - e)$ or from distribution of e_{sm}, depending on whether $\mathcal{C}$ provides real or simulated partial decryptions to $\mathcal{A}$. Let δ and δ' denote the advantages with which $\mathcal{A}$ distinguishes $D_0^{\eta}(r)$ from $D_1^{\eta}(r)$ in the presence of real or simulated partial decryptions respectively. Assuming that the aforementioned distinguishing problems are "publicly sampleable" [BLRL+18], the relation below follows from known results in [BLRL+18]: $\delta' \geq \frac{\delta}{4R_a(\Psi||\Psi')} \cdot \left(\frac{\delta}{2}\right)^{\frac{a}{a-1}}$, where Ψ and Ψ' denote the distribution of $(e_{sm} - e)$ and the distribution of e_{sm} respectively and $R_a(\Psi||\Psi')$ is the Rényi divergence of order a between the distributions Ψ and Ψ'.

Arguing Public Sampleability. In order to invoke the aforementioned relation, we first need to argue that the aforementioned distinguishing problems satisfy the notion of public sampleability as defined in [BLRL+18]. Given a bit $b' \in \{0,1\}$ and a sample x from $D_b^{\eta}(r)$, we can publicly sample a fresh element x' of $D_{b'}^{\eta}(r)$ by (i) replacing $\mathbf{CT}_b$ of x with $\mathbf{CT}_{b'} = \{\mathsf{TRLWE.Enc}(\mathbf{PK}, M_j^{b'})\}_{j\in[\ell]}$, (ii) replacing $\{\widehat{\mathsf{CT}}_i^{b}\}_{i\in[Q]}$ with $\{\widehat{\mathsf{CT}}_i^{b'}\}_{i\in[Q]} = \{\mathsf{TRLWE.Eval}(\mathbf{PK}, C_i, \mathbf{M}_{b'})\}_{i\in[Q]}$ and, (iii) replacing the last component $\{p_i^b\}_{i\in[Q]}$ with $\{p_i^{b'}\}_{i\in[Q]}$ such that $p_i^{b'} = \gamma + r_i$. Here, computation of γ requires the knowledge of $\widehat{CT}_i^{b'}$. Thus we can publicly generate a new sample x' from distribution $D_{b'}^{\eta}(r)$. Hence $D_b^{\eta}(r)$ is indeed publicly sampleable.

Completing the Proof. We can now invoke known results from [TT15] and the *multiplicative property* of Rényi Divergence to argue that for any $a \in (1,\infty)$, we have,

$$R_a(\Psi||\Psi') \leq \exp\left(\frac{a \cdot \pi \cdot N \cdot \|e\|_\infty^2}{\sigma^2}\right),$$

where $\|e\|_\infty$ denotes the infinity norm of the degree $(N-1)$-RLWE noise polynomial e. Assuming that $\|e\|_\infty \leq c\alpha$, where c is some constant and α is the standard deviation of Gaussian RLWE noise distribution $\mathcal{G}$, we have, $R_a(\Psi||\Psi') \leq \exp\left(\frac{a\cdot\pi\cdot N\cdot c^2\cdot\alpha^2}{\sigma^2}\right)$.

Finally, for the scenario where the adversary $\mathcal{A}$ sees a maximum of $Q = poly(\lambda)$ such partial decryption samples, we invoke the multiplicative properties of Rényi Divergence to state the following:

$$R_a(\Psi||\Psi') \leq \exp\left(\frac{a\cdot\pi\cdot Q\cdot N\cdot c^2\cdot\alpha^2}{\sigma^2}\right).$$

Parameter Choices (Lower Bounds). At this point, we are ready to propose the asymptotic parameter choices for our ThFHE scheme TRLWE supporting (T, T)-threshold decryption. Assume that the adversary $\mathcal{A}$ sees at most $Q = \text{poly}(\lambda)$ partial decryption samples, let σ and α be the standard deviation parameters for the Gaussian distributions pertaining to the smudging noise and RLWE noise, respectively, and let c be a constant such that $|e| \leq c\alpha$ (e being the RLWE noise polynomial). It suffices for us to choose σ such that

$$\sigma \geq c\cdot\alpha\cdot\sqrt{Q\cdot N},$$

since this yields $R_a(\Psi||\Psi') \leq \exp(a\cdot\pi)$, and hence

$$\delta' \geq \frac{\delta}{4}\cdot\left(\frac{\delta}{2}\right)^{\frac{a}{a-1}}\cdot\exp(-a\cdot\pi).$$

Taking any value of $a > 1$ yields the desired condition on δ and δ', i.e., non-negligible δ would result in non-negligible δ'. Note that it suffices for σ to be *only polynomially larger* than α. Hence, our scheme is secure whenever $\sigma \geq c\cdot\alpha\cdot\sqrt{Q\cdot N}$.

Parameter Choices (Upper Bounds). It remains to answer the question of *upper bounding* the amount of smudging noise that each party can add, and here we allow the maximum possible smudging noise that does not affect the correctness of the distributed decryption protocol. Formally, let $q = 2^{\lambda_1}$ be the TRLWE modulus (or equivalently, suppose that the Torus-FHE scheme supports a maximum precision of λ_1 bits) and let $p = 2^{\lambda_2}$ be the size of the space of message-polynomial coefficients (or equivalently, suppose that the Torus-FHE scheme supports message-polynomial coefficients with a precision of λ_2 bits) such that $p \leq q$. At a high level, to ensure the correctness of (T, T)-distributed decryption, we need the *total* noise to be upper bounded by $\Delta/2$, where $\Delta = q/p = 2^{\lambda_1-\lambda_2}$. More formally, for correctness of (T, T)-distributed decryption to hold, we must have $\|e\|_\infty + T\cdot\|e_{sm}\|_\infty < \Delta/2$, where $\|\cdot\|_\infty$ denotes the infinity norm of some polynomial. Since $\|e_{sm}\|_\infty > \|e\|_\infty$ (by the lower bound argument presented above), we choose

$$\|e_{sm}\|_\infty < \Delta/2(T+1).$$

Combined with the lower bounds imposed by the Rényi divergence-based security analysis presented earlier, we thus avoid the super-polynomial modulus-to-noise ratio (ratio between modulus q and any coefficient of RLWE noise polynomial e) incurred by all prior works on ThFHE, thereby yielding the *first practical* ThFHE *scheme with polynomial modulus-to-noise ratio.*

4.6 Analysis of Generalized (t, T)-threshold FHE

In this section, we prove the correctness and security of our proposed (t, T)-threshold FHE scheme. Concretely, we prove that our proposed (t, T)-threshold decryption mechanism satisfies the notion of IND-security described in Section 3.3 (adopted from [JRS17]). Our security analysis uses a generalized version of our Rényi divergence-based proof argument outlined earlier in Section 4.5. This section also presents a detailed discussion on the asymptotic choice of noise parameters for our proposed scheme, with upper bounds on the smudging noise imposed by our correctness analysis, and (more crucially) polynomial lower bounds on the smudging noise derived from our Rényi divergence-based security proof.

4.6.1 Correctness of (t, T)-Threshold Decryption

In this section, we discuss the upper bounds for the different noise parameters in order to ensure the correctness of our proposed (t, T)-threshold decryption procedure.

Some Notations. Let us assume $q = 2^{\lambda_1}$ to be the modulus in TRLWE and $|\mathcal{M}| = p = 2^{\lambda_2}$ to be size of the space of coefficients of message-polynomial such that $p \leq q$. Now, let $\Delta = \frac{q}{p} = 2^{\lambda_1 - \lambda_2}$ denote the distance between two consecutive value of a message coefficient in $\mathcal{M}$. Note that we assume $\Delta = 1$ throughout the paper, as in Torus-FHE library $\lambda_1 = \lambda_2 = 32$ have been considered.

TRLWE noise. When applying $\mathsf{TRLWE.Decode}_0$ on a TRLWE ciphertext $\mathsf{CT} = (A, B)$, we effectively compute $\Phi = B - A \cdot \mathbf{SK}$, which essentially equals $\Delta \cdot \mathsf{m} + e$. Now, Δ being a constant we can rewrite Φ as $\sum_{i=0}^{N-1} (\Delta \cdot \mathsf{m}_i + e_i) x^i$. Next, we round up and approximate each coefficient of Φ during $\mathsf{TRLWE.Decode}_1$ as $\Delta \cdot \mathsf{m}_i + e_i \xrightarrow{round} \Delta \cdot \mathsf{m}_i \xrightarrow{approximate} \mathsf{m}_i$. For correctness, we need $|e_i| < \Delta/2$.

Smudging noise. The parties eventually combine their own partial decryptions in order to compute an unmasking component $part\_decrypt_f$. Without loss of generality, for party P_1, $part\_decrypt_f$ is computed as:

$$\left(A \cdot SH_1 - \sum_{j=2}^{t} A \cdot SH_j + e_{sm}^1 - \sum_{j=2}^{t} e_{sm}^j \right).$$

The final message recovery step proceeds as:

$$B - part\_decrypt_f = \Delta \cdot \mathsf{m} + e - \left(e_{sm}^1 - \sum_{l=2}^{t} e_{sm}^l \right).$$

Here, $e = e_0 + e_1 x + \cdots + e_{N-1} x^{N-1}$ is the RLWE noise polynomial and $e_{sm}^i = e_{sm,0}^i + e_{sm,1}^i x + \cdots + e_{sm,N-1}^i x^{N-1}$ is the smudging noise polynomial added by party P_i. Hence, for correct decryption, the following condition should hold for each $z \in [0, N-1]$:

$$|e_z - e_{sm,z}^1 + \sum_{l=2}^{t} e_{sm,z}^l)| < \frac{\Delta}{2}.$$

Let $\|e\|_\infty$ and $\|e_{sm}\|_\infty$ denote the infinity norms of the RLWE noise polynomial e and the smudging noise polynomial e_{sm}, respectively. Then, we must have:

$$\|e\|_\infty + t \cdot \|e_{sm}\|_\infty < \Delta/2.$$

Since $\|e_{sm}\|_\infty > \|e\|_\infty$ (by the lower bound argument presented above), it suffices to choose $\|e_{sm}\|_\infty < \Delta/2(t+1)$.

4.6.2 IND-Security of (t, T)-Threshold Decryption

Let us recall the (t, T)-threshold decryption technique of Section 4.4 for TRLWE, a Torus-FHE instantiation introduced in Section 4.1. Also, recall the notion of security in the form of a game between adversary $\mathcal{A}$ and challenger $\mathcal{C}$ from Section 3.3 with respect to our proposed threshold FHE scheme TRLWE.

A Close Look at Partial Decryptions. First, we take a close look at the partial decryption component returned by the challenger $\mathcal{C}$ in the partial decryption query phase of the security game. We allow the corrupted subset $\mathcal{S}$ in the security game to be of maximal size, i.e., $|\mathcal{S}| = (t-1)$. Let us assume $\widehat{\mathsf{CT}}_i$ to be the i^{th} evaluated ciphertext during partial decryption query phase, i.e., $\widehat{\mathsf{CT}}_i$ is obtained by homomorphically evaluating C_i on the set of inputs $\mathbf{CT}^\star = \{\mathsf{CT}_j^\star\}_{j\in[\ell]}$ for some $i \in [Q]$. Here, $\mathbf{CT}^\star = \{\mathsf{CT}_j^\star\}_{j\in[\ell]}$ is the challenge set returned to the adversary in the challenge phase of the security game. A partial decryption of $\widehat{\mathsf{CT}}_i = (\hat{A}_i, \hat{B}_i)$ by some honest party $P_j \notin \mathcal{S}$ corresponds to a t-sized group $P_j \bigcup \mathcal{S}$ with group_id g and we denote it with $p_{i,j}$. Challenger $\mathcal{C}$ computes $p_{i,j}$ as follows,

$$p_{i,j} = \hat{A}_i \cdot SH_{j,g} + e_{sm},$$

where $SH_{j,g}$ is P_j's secret share corresponding to group_id g and e_{sm} is a smudging noise polynomial with each coefficient sampled from $\mathcal{G}_{sm}$. However using the linear reconstruction property of Benaloh-Leichter $\{0,1\}$-LISSS (Section 4.3), we can alternatively express $p_{i,j}$ as following,

$$p_{i,j} = \hat{B}_i - m_i - e - \sum_{P_k \in \mathcal{S}} \hat{A}_i \cdot SH_{k,g} + e_{sm},$$

where e is the RLWE noise polynomial of ciphertext $\widehat{\mathsf{CT}}_i$ with each of its coefficients sampled from $\mathcal{G}$, and m_i is the expected output of the circuit C_i in plaintext, i.e., $C_i(m_1^0, \ldots, m_\ell^0) = C_i(m_1^1, \ldots, m_\ell^1) = m_i$. Letting $\gamma_i = \hat{B}_i - m_i - \sum_{P_k\in\mathcal{S}} \hat{A}_i \cdot SH_{k,g}$, we get

$$p_{i,j} = \gamma_i + r_{i,j},$$

such that $r_{i,j}$ is sampled from distribution of $(e_{sm} - e)$.

We can publicly simulate the partial decryption by some honest party $P_j \notin \mathcal{S}$ as follows,

$$p'_{i,j} = \gamma_i + r'_{i,j},$$

where $r'_{i,j}$ is sampled from distribution of e_{sm}.

Defining Some Distributions. Recall the security game between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$ in Section 3.3 with respect to our proposed scheme TRLWE. Let η is the set of some fixed parameters in a particular instance of the game as follows,

$$\eta = (\mathbf{PK}, \mathbf{SK}, \{\mathbf{SK}_i\}_{i\in[T]}, \mathcal{S}, \mathbf{M}_0, \mathbf{M}_1, \{C_i\}_{i\in[Q]}).$$

Algorithm 2 Public Sampling Algorithm PS

Input:

$$x = (\mathbf{PK}, \{\mathbf{SK}_i\}_{P_i \in \mathcal{S}}, \mathbf{M}_0, \mathbf{M}_1, \mathbf{CT}_b, \{\widehat{\mathsf{CT}}_i^b\}_{i \in [Q]}, \{p_{i,j}^b\}_{i \in [Q], P_j \notin \mathcal{S}}), b' \in \{0,1\}.$$

Output: $x' \in D_{b'}^{\eta}(r)$.

1: Compute a new challenge set $\mathbf{CT}_{b'}$, which is component-wise fresh encryption of $\mathbf{M}_{b'}$.
2: For each $i \in [Q]$, generate an evaluated ciphertext $\widehat{\mathsf{CT}}_i^{b'} = (\hat{A}_i^{b'}, \hat{B}_i^{b'}) = \mathsf{TRLWE.Eval}(\mathbf{PK}, C_i, \mathbf{CT}_{b'})$.
3: For each $i \in [Q]$, first compute $\gamma_i = \hat{B}_i^{b'} - m_i - \sum_{P_k \in \mathcal{S}} (\hat{A}_i^{b'} \cdot SH_{k,g})$ and then for each honest party $P_j \notin \mathcal{S}$, compute partial decryption $p_{i,j}^{b'} = \gamma_i + r_{i,j}$. Here g is $\mathsf{group\_id}$ of $P_j \bigcup \mathcal{S}$.
4: Return a fresh sample x' as,

$$x' = (\mathbf{PK}, \{\mathbf{SK}_i\}_{P_i \in \mathcal{S}}, \mathbf{M}_0, \mathbf{M}_1, \mathbf{CT}_{b'}, \{\widehat{\mathsf{CT}}_i^{b'}\}_{i \in [Q]}, \{p_{i,j}^{b'}\}_{i \in [Q], P_j \notin \mathcal{S}}).$$

Let $r = \{r_{i,j}\}_{i \in [Q], P_j \notin \mathcal{S}}$ be a set of noise parameters. We define the distribution $D_b^{\eta}(r)$ parameterized by η as follows,

$$D_b^{\eta}(r) = (\mathbf{PK}, \{\mathbf{SK}_i\}_{P_i \in \mathcal{S}}, \mathbf{M}_0, \mathbf{M}_1, \mathbf{CT}_b, \{\widehat{\mathsf{CT}}_i^b\}_{i \in [Q]}, \{p_{i,j}^b\}_{i \in [Q], P_j \notin \mathcal{S}}).$$

Here each component is analogous to the components described in $\mathsf{G}_{\mathsf{ThFHE}, \mathcal{A}, \mathbb{A}_{t,T}}(1^\lambda, 1^d)$ of Section 3.3. Component $\mathbf{CT}_b$ of $D_b^{\eta}(r)$ denotes the scenario when the challenge set $\mathbf{CT}^\star$ in the game is component-wise encryption of $\mathbf{M}_b$, i.e., $\mathsf{CT}_j^\star = \mathsf{TRLWE.Enc}(\mathbf{PK}, M_j^b)$ for each $j \in [\ell]$.

Public Sampleability of $D_b^{\eta}(r)$. We argue the public sampleability [BLRL+18] of $D_b^{\eta}(r)$ by providing a public sampling algorithm PS in Algorithm 2. Given any sample

$$x = (\mathbf{PK}, \{\mathbf{SK}_i\}_{P_i \in \mathcal{S}}, \mathbf{M}_0, \mathbf{M}_1, \mathbf{CT}_b, \{\widehat{\mathsf{CT}}_i^b\}_{i \in [Q]}, \{p_{i,j}^b\}_{i \in [Q], P_j \notin S}),$$

from $D_b^{\eta}(r)$ with unknown bit b and a bit b', it generates fresh sample

$$x' = (\mathbf{PK}, \{\mathbf{SK}_i\}_{P_i \in \mathcal{S}}, \mathbf{M}_0, \mathbf{M}_1, \mathbf{CT}_{b'}, \{\widehat{\mathsf{CT}}_i^{b'}\}_{i \in [Q]}, \{p_{i,j}^{b'}\}_{i \in [Q], P_j \notin \mathcal{S}}),$$

of $D_{b'}^{\eta}(r)$ efficiently.

Note that the algorithm requires the knowledge of $\mathbf{PK}$, the circuits $\{C_i\}_{i \in [Q]}$, the output of the circuit evaluations in plaintext (i.e., $\{m_i\}_{i \in [Q]}$, which is independent of bit b), the noise samples $\{r_{i,j}\}_{i \in [Q], P_j \notin \mathcal{S}}$ and the secret shares of the corrupted parties in $\mathcal{S}$. All these information are publicly available and hence we can indeed publicly generate a valid sample x' of $D_{b'}^{\eta}(r)$ efficiently. Hence we conclude that the distribution $D_b^{\eta}(r)$ is publicly sampleable.

Proof Outline. Notice that $D_b^{\eta}(r)$ captures the view of the adversary in the security game when the challenger $\mathcal{C}$ samples the bit b in the challenge phase and $r = \{r_{i,j}\}$ is the set of noise values sampled from the distribution of $e_{sm} - e$ and used in computing partial decryptions $\{p_{i,j}\}$ for all $i \in [Q]$ and $P_j \notin \mathcal{S}$. However in simulated world, we can simulate partial decryptions $p'_{i,j} = \gamma_i + r'_{i,j}$, by just sampling each $r'_{i,j}$ from the distribution of e_{sm}. Let us denote a problem P as distinguishing a sample of $D_0^\eta(r)$ from a sample of $D_1^\eta(r)$ and the problem P' as distinguishing a sample of $D_0^\eta(r')$ from a sample of $D_1^\eta(r')$. Using a novel Rényi divergence based analysis, we show that non-negligible distinguishing advantage of problem P leads to non-negligible distinguishing advantage of problem P'. But due to hardness of binary ring-LWE problem, no PPT adversary can distinguish $\mathbf{CT}_0$ from $\mathbf{CT}_1$ in the simulated world, as it gains no effective information about the actual secret shares of honest parties by seeing the simulated partial decryptions. Thus distinguishing advantage of problem P' is already known to be negligible due to binary ring-LWE assumption. Now by contradiction we conclude that the distinguishing advantage in P is negligible, making our TRLWE a secure ThFHE scheme.

Rényi Divergence based Analysis. Recall from Theorem 4.2 of [BLRL+18], due to public sampleability property of $D_b^\eta(r)$, if there exists a τ-time distinguisher $\mathcal{D}$ for problem P with distinguishing probability δ, then there must exists a distinguisher $\mathcal{D}'$ for Problem P' with distinguishing probability δ' with run-time τ', such that,

$$\delta' \geq \frac{\delta}{4R_a(\Psi||\Psi')} \cdot \left(\frac{\delta}{2}\right)^{\frac{a}{a-1}},$$

$$\tau' \leq \frac{64}{\delta^2}\log(\frac{8R_a(\Psi||\Psi')}{\delta^{a/(a-1)+1}})(\tau_S + \tau).$$

Here, τ_S is the run-time of public sampling algorithm for $D_b^\eta(r)$. Also Ψ is the distribution of $(e_{sm} - e)$ and Ψ' is the distribution of e_{sm}. Now, with the results from Lemma 5 in [TT15] and *multiplicative property* of Rényi Divergence we argue that for any $a \in (1, \infty)$:

$$R_a(\Psi||\Psi') \leq \exp\left(\frac{a \cdot \pi \cdot N \cdot \|e\|_\infty^2}{\sigma^2}\right),$$

where $\|e\|_\infty$ denotes the infinity norm of the $(N-1)$-degree RLWE noise polynomial e. Assuming that $\|e\|_\infty \leq c\alpha$, where c is some constant and α is the standard deviation of RLWE noise distribution $\mathcal{G}$, we have

$$R_a(\Psi||\Psi') \leq \exp\left(\frac{a \cdot \pi \cdot N \cdot c^2 \cdot \alpha^2}{\sigma^2}\right).$$

Finally, considering the scenario that the adversary $\mathcal{A}$ does $Q = poly(\lambda)$ number of queries, and thus is able to see a total of $Q \cdot (T - t + 1)$ number of partial decryptions corresponding to $(T - t + 1)$ number of honest parties, we invoke the multiplicative properties of Rényi Divergence from [TT15] to state the following:

$$R_a(\Psi||\Psi') \leq \exp\left(\frac{a \cdot \pi \cdot Q \cdot (T - t + 1) \cdot N \cdot c^2 \cdot \alpha^2}{\sigma^2}\right).$$

Observe that it suffices for us to choose σ such that

$$\sigma \geq c \cdot \alpha \cdot \sqrt{Q \cdot (T - t + 1) \cdot N},$$

since this yields $R_a(\Psi||\Psi') \leq \exp(a \cdot \pi)$, and hence:

$$\delta' \geq \frac{\delta}{4} \cdot \left(\frac{\delta}{2}\right)^{\frac{a}{a-1}} \cdot \exp(-a \cdot \pi) = \frac{1}{2} \cdot \left(\frac{\delta}{2}\right)^{\frac{2a-1}{a-1}} \cdot \exp(-a \cdot \pi).$$

and for the run-time we have,

$$\tau' \leq \frac{64}{\delta^2} \log(\frac{8 \cdot \exp(a \cdot \pi)}{\delta^{a/(a-1)+1}})(\tau_S + \tau).$$

Hence the condition $\sigma \geq c \cdot \alpha \cdot \sqrt{Q \cdot (T - t + 1) \cdot N}$ (i.e., smudging noise is only polynomially larger than RLWE noise) implies that, for any $a > 1$, non-negligible δ would result in non-negligible δ'. This completes the proof of security for our proposed TRLWE scheme supporting (t, T)-threshold decryption.

5 Software Implementation and Experimental Evaluation

We now describe a prototype implementations of our (t, T)-threshold decryption scheme over Torus-FHE on two extreme varieties of computing platforms - a high-end x86-based server, and a low-end resource-constrained ARM-based platform[1]. We report the implementation and performance of TRLWE, a public-key adaptation of the ThFHE scheme built upon Torus-FHE library equipped with our proposed threshold decryption mechanism. Although Torus-FHE library implements a symmetric-key version of the underlying FHE scheme, we use the idea of [Rot11] to implement its public-key version and extend it to support threshold decryption in order to get the desired implementation of TRLWE. We stress that this is, to the best of our knowledge, the first practical implementation of any ThFHE scheme. In particular, a concurrent work [BS23] that *theoretically* proposes a threshold FHE scheme with polynomial modulus-to-noise ratio, do not report any implementation or performance results to the best of our knowledge.

In our setting, the threshold secret sharing is done by a trusted cloud server with sufficient computational resources. Subsequently, homomorphic evaluations also happen on encrypted data stored at the cloud server. The key focus of our implementation is in realizing the proposed threshold decryption algorithm on resource-constrained handheld devices; hence our experiments and evaluation focus purely on the performance of our threshold decryption implementation.

For the sake of completeness, we implement our threshold decryption algorithm on two kinds of platforms, lying at two extreme ends of the spectrum of computational capabilities: (a) a high-end workstation with an Intel(R) Xeon(R) CPU E5-2690 v4 CPU (2.60GHz clock-frequency), 28 physical cores, and 128GB RAM, and (b) a low-end Raspberry Pi 3b board with a Quad Core 1.2GHz Broadcom BCM2837 64bit CPU and 1 GB RAM running Raspberry Pi OS Lite (Linux kernel version: 5.10.63-v7+).

Our first implementation is optimized for high performance and, as verified by our experiments, yields extremely fast threshold decryption times. Our second implementation is optimized for extracting maximum performance out of a low-end resource-constrained platform, and yields reasonably practical threshold decryption times. Before describing our evaluation, we present some more details of our implementation.

[1] Our implementation code and additional (low-level) implementation details are available at: `https://anonymous.4open.science/r/ThFHE_artifacts-2FD3`

5.1 Implementation Details

We implement the natural public-key analogue of the Torus-FHE library which originally implements a secret-key version, while leaving implementation of the homomorphic evaluation unchanged. This makes our implementation cross-compatible with other libraries (e.g., NuFHE) that build directly upon Torus-FHE. Since our core contribution lies in thresholdizing the decryption process, which only requires the secret key, we keep our discussion limited to generating and sharing the secret key.

We extend the Torus-FHE library to support threshold key generation and threshold decryption. We use the Torus-RLWE secret key generation routine to generate the secret key with a set of parameters that is chosen by relying on our proposed Rényi divergence-based security argument (see Section 4.6.1 for the detailed analysis). In particular, this analysis enables a polynomial modulus-to-noise ratio, which crucially allows our implementation to be practically deployable on a resource-constrained platform.

Once the key has been generated, we build the distribution matrix M and share matrix ρ (see Section 4.3). The distribution matrix generation, when implemented directly in software, results in a recursive implementation, which potentially results in high memory access overheads, and is unsuitable for resource-constrained platforms. However we can avoid these excess function call overheads and generate it iteratively in one go by exploiting a regular pattern, as discussed in Appendix A.

For the partial and threshold decryption functions, we have two implementations. The first implementation targets a high-end processor, and directly leverages Torus-FHE APIs for fast polynomial multiplication using Fast Fourier Transform (FFT), as is required in the partial decryption phase. The other is a portable implementation suited for low-end resource-constrained handheld devices. In particular, the latter replaces the FFT polynomial multiplication, which depends on x86 AVX instructions for efficiency, with a naïve schoolbook multiplication. This is done to keep the implementation as architecture-agnostic and lightweight as possible. In the porting process, we have removed multiple dynamic memory allocation steps to achieve better memory efficiency. Also, our observation that each of the participating parties except one receives a binary key share through (t, T)-threshold secret sharing significantly contributes to reduce the cycle counts in polynomial multiplication in both implementations.

5.2 Experimental Evaluation

In this section we validate our proposed threshold decryption technique by an implementation over Torus-FHE library and the steps involved are summarized in Algorithm 3. The output of BootstrappedOR in step 3 here is a Torus-LWE ciphertext and we convert it to a Torus-RLWE ciphertext in step 4, in order to support packing of multiple plaintext bits together.

In accordance with our intended use-case, we experimentally evaluate steps 1 through 5 of Algorithm 3 on a high-end server, and step 6 on both the high-end server and a low-end resource-constrained handheld device. In particular, in our experiments we measure the time taken by steps 5 and 6. Note that step 6 includes both partial decryption and final combination. Table 1 lists the concrete parameters used in our experiments.

Algorithm 3 Software implementation of cryptosystem with (t, T)-threshold decryption

Input: $inp_1 \in \mathbb{N}$, $inp_2 \in \mathbb{N}$, $t \in \mathbb{N}$, $T \in \mathbb{N}$, $\mathcal{P} \subset [1, T]$ s.t $|\mathcal{P}| = t$
Output: $outp \leftarrow inp_1 \vee inp_2$
1: $(LweSK, LwePK) \leftarrow$ LweKeyGen
2: $cipher_1 \leftarrow$ Encrypt$(LwePK, inp_1)$, $cipher_2 \leftarrow$ Encrypt$(LwePK, inp_2)$
3: $eval\_res \leftarrow$ BootstrappedOR$(cipher_1, cipher_2, LwePK)$
4: $(ring\_cipher, RLweSK) \leftarrow$ ConvertLWEtoRLWE$(eval_res, LweSK)$
5: ShareSecret$(RLweSK, t, T)$ ▷ Now all parties get their key shares. Each party $i \in \mathcal{P}$ calculates $outp$ on its own.
6: $outp \leftarrow$ ThresholdDecrypt$(ring_cipher, \mathcal{P}, t, T, i)$
7: **return** $outp$

Table 1: Parameters used in experimental setting

Parameter	**Value**
k (Number of polynomials in Torus-LWE ciphertext)	1
n (Torus-LWE dimension)	1024
N (Degree of Torus-RLWE polynomial is $(N-1)$)	1024
α (Standard deviation of Torus-RLWE noise)	2^{-25}
σ (Standard deviation of smudging noise)	2^{-6}

The choices for n, N and α are compatible with the Torus-FHE library. For smooth conversion in step 4 of Algorithm 3, we fix $k = 1$. We choose the last parameter based on a lower-bound given by our Rényi divergence based analysis and an upper-bound imposed by correctness.

Figures 1 and 2 show the secret key sharing time, the partial decryption time, the final decryption time, and the plain decryption time on a high-end workstation in terms of milliseconds and clock cycles respectively, while Figures 3 and 4 show the partial and final decryption times in milliseconds and clock cycle counts respectively on the low-end Raspberry Pi 3b platform.

The partial decryption time in all the figures follow a constant trend as in our use case, it is done parallelly in individual devices and the vector or polynomial sizes do not change with the number of parties. We emphasize that, as a direct consequence of the efficient parameter choices for threshold FHE enabled by our Rényi Divergence-based analysis, the threshold decryption timing is practical even on a highly resource-constrained ARM-based platform.

Finally, Figure 5 shows that the time required for threshold decryption[1] is only slightly higher than that of plain decryption using the Torus-FHE library, for both the high-end workstation and the resource constrained device. In other words, our proposed threshold decryption procedure incurs only minimal overhead over the plain decryption algorithm specified in the original Torus-FHE library. To the best of our knowledge, this is the

[1]We report the *end-to-end* threshold decryption time, wherein we add up the time for a single partial decryption (since the partial decryption phase is meant to be done in parallel by each participating party) and the time for the final re-combination of partial decryptions. Also, the overall threshold decryption time is dominated by the former component, which is independent of (t, T), and hence, the overall threshold decryption time in Figure 5 grows only minimally with increasing (t, T) values.

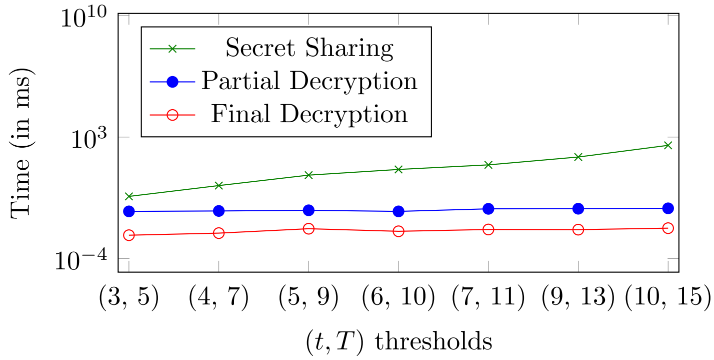

Figure 1: Secret Sharing and Threshold Decryption Time in High-End Server. Note that the y-axis is in logarithmic scale.

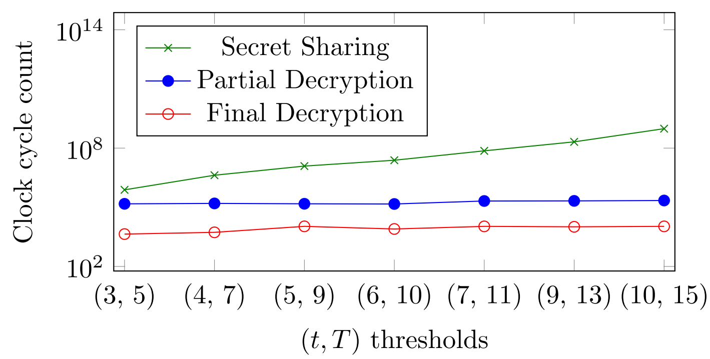

Figure 2: Secret Sharing and Threshold Decryption Clock Cycles in High-End Server

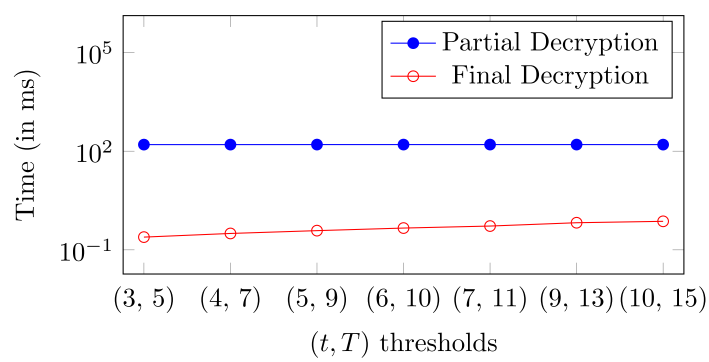

Figure 3: Threshold Decryption Time in Handheld device

first prototype of threshold FHE with the capability of executing the threshold decryption algorithm practically on resource-constrained platforms.

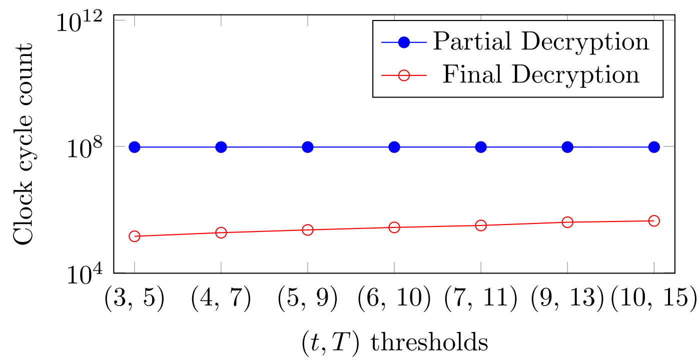

Figure 4: Threshold Decryption Clock Cycles in Handheld device

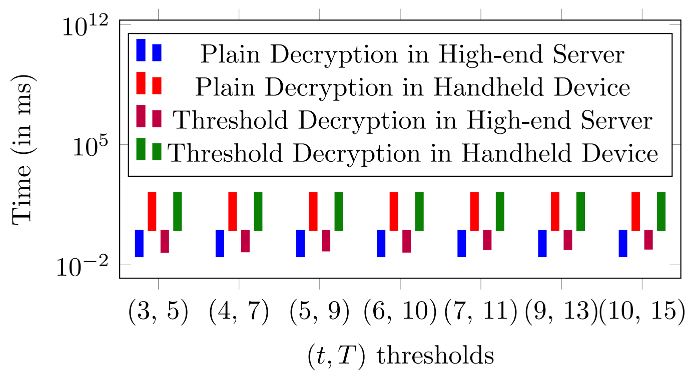

Figure 5: Time comparison between Plain Decryption and Threshold Decryption

6 Case-Study: Computing over Encrypted Medical Data

In this section, we use our proposed ThFHE scheme over the Torus to realize an end-to-end usecase of outsourced computations over encrypted medical datasets, where the final outcome is computed in a distributed manner by multiple entities (e.g. doctors, research laboratories, or other medical practitioners). Concretely, we illustrate the efficacy of our proposal via experiments evaluating encrypted computations over a real medical database, as well as distributed decryptions of the computed result on resource-constrained handheld devices, where both the encryption and distributed decryption operations are performed using our proposed ThFHE scheme. We perform an encrypted K-Nearest Neighbours (KNN) classification [SCK14] that outputs an encrypted prediction bit indicating the possibility of cardiovascular disease. The classification is done based on a patient's encrypted medical records and pre-computed encrypted training data.

6.1 Encrypted KNN Computation

The encrypted KNN algorithm (Algorithm 4) takes as input: (a) an encrypted set of test data, which is to be predicted, (b) an encrypted set of training data to train the KNN algo-

Algorithm 4 KNN over encrypted medical data

Input: $\boldsymbol{test\_data} = \text{Encrypt}(k, test\_patient)$,
$\boldsymbol{train\_data} = \{\text{Encrypt}(k, patient_1), \ldots, \text{Encrypt}(k, patient_n)\}$,
$\boldsymbol{bk}$ = bootstrapping key, K = KNN parameter.
Output: $decisional\_bit = \text{Encrypt}(k, predicted\_bit)$

1: Initialize a Torus-LWE ciphertext array *distant* of size n
2: **for** $i = 1$ **to** n **do**
3: $\quad distant[i] \leftarrow \text{Manhattan}(test\_data, train\_data[i], \boldsymbol{bk})$
4: $sorted\_train\_data \leftarrow \text{BubbleSort}(distant, train\_data, \boldsymbol{bk})$
5: Initialize a counter ciphertext $\boldsymbol{count} = \text{Encrypt}(k, zero)$ to count the decision of K-Nearest Neighbours
6: **for** $i = 1$ **to** K **do**
7: $\quad count \leftarrow count + \text{Decision}(sorted\_train\_data[i]))$
8: Initialize a Torus-LWE variable $threshold = \text{Encrypt}(k, \lfloor K/2 \rfloor)$
9: $decisional\_bit \leftarrow \text{Difference}(threshold, count, bk)$
10: **return** *decisional_bit*

rithm, (c) the bootstrapping key $\boldsymbol{bk}$, which is a part of public key and (d) KNN parameter K to output an encrypted single prediction bit. Following the approach outlined in [RC19], we sub-divide the encrypted KNN computation algorithm into three parts as described below.

Encrypted Manhattan Distance Computation. First, the encrypted Manhattan distances between the testing data and all the training data are homomorphically computed and stored in the *distant* variable. The Manhattan distance is preferred over other distances to avoid the "curse of dimensionality" problem in machine learning [AHK01]. To compute the difference between two ciphertexts (ThFHE_{DIFF} between $\text{Encrypt}(k, Plain_1)$ and $\text{Encrypt}(k, Plain_2)$), we use the 2's complement form representation.

Sorting over Encrypted Data. In this step, the neighbours are sorted in ascending order based on the calculated distances. The encrypted-bubble-sort implementation directly uses encrypted-comparison and sorting techniques from prior-works [RC19, CSS20, CS20, ÇDSS15]. Our encrypted bubble sort implementation takes the bootstrapping key, the patient's encrypted data and their corresponding encrypted Manhattan distances, and outputs the sorted patient data based on these encrypted distances.

Prediction over Encrypted Data. The encrypted decision bits of KNN computation are added to get $\text{Encrypt}(k, count)$ (line 7, Algorithm 4), which is then compared homomorphically with the threshold value ($\text{Encrypt}(k, \lfloor K/2 \rfloor)$) to arrive at the (encrypted) decision. The final plaintext decision is recovered via threshold decryption.

6.2 Experimental results

We consider a cardiovascular disease related dataset[1] that comprises of 70000 data instances and 12 features like *age_days*, height, weight, *ap_lo* (Diastolic Blood Pressure), *ap_hi* (Systolic Blood Pressure), gender, cholesterol, glucose, smoke, alcohol, active, id, and cardio.

[1] https://www.kaggle.com/sulianova/cardiovascular-disease-dataset

Table 2: Encrypted KNN execution time

K	**Neighbour size**	**Prediction time without OpenMP (in minutes)**	**Prediction time with OpenMP (in minutes)**	**Prediction time with OpenMP (cycle count in 10^{12})**
5	10	154.88	15.96	2.49
	20	369.90	29.71	4.64
	30	558.36	49.96	7.79
	40	850.70	60.11	9.38
	50	1062.86	72.90	11.37
7	10	194.81	18.50	2.89
	20	431.41	33.66	5.25
	30	726.13	50.08	7.81
	40	975.01	63.83	9.96
	50	1282.08	79.78	12.45
9	10	235.30	24.23	3.78
	20	527.98	46.66	7.28
	30	844.36	60.55	9.45
	40	1146.26	80.81	12.61
	50	1498.43	99.31	15.49

Out of 12 features, cardio is the target feature which need to be predicted based upon the rest 11 features. An accuracy of 70% has been achieved with 60 (training=40, testing =20) data-rows and that can be improved further by performing more hyperparameter tuning and by incorporating more data-rows into account. Out of 70000 data instances, we randomly select 60 data instance and divided it into different training is to testing ratio. Table 2 shows the execution time of KNN algorithm in two variants, with (using OpenMP) and without any parallel processing techniques. The OpenMP version has big advantage of parallelizing multiple loops to facilitate smaller execution time as shown in Table 2 for $K = 5, 7$, and 9. Note that the number of OpenMP threads used during each execution is equal to the neighbour size listed in Table 2. The execution platform is equipped with Intel(R) Xeon(R) CPU E5-2690 v4 with 2.60GHz clock. The system has 132GB of RAM and 56 available physical core.

7 Conclusion and Future Work

We presented the design, analysis and practical implementation for a novel threshold FHE scheme from the hardness of Binary Ring-LWE with polynomial modulus-to-noise ratio. We showed, for the first time, that threshold FHE can actually be deployed in a fast, scalable and reasonably resource-efficient manner for real-time applications via benchmarking experiments on two extreme varieties of computing platforms - a high-end x86-based server and a low-end resource-constrained ARM-based platform. We showcased an end-to-end implementation of our proposed system and used it for fast, scalable yet secure k-nearest-neighbor computations over encrypted medical data outsourced to a cloud service provider.

Our work gives rise to many interesting directions of future research. In particular, we leave it as an open question to extend our Rényi divergence-based security analysis techniques to the setting of multi-key FHE with threshold decryption, for which all known realizations still require super-polynomial modulus-to-noise ratio. Such an extension would enable efficient realizations of richer applications such as round-optimal multi-party computation.

References

[AHK01] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory — ICDT 2001*, pages 420–434, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[AJJM20] Prabhanjan Ananth, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Multi-key fully-homomorphic encryption in the plain model. In *TCC 2020*, volume 12550, pages 28–57, 2020.

[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 David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012.

[ASY22] Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. *Cryptology ePrint Archive*, 2022.

[BBH06] Dan Boneh, Xavier Boyen, and Shai Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In David Pointcheval, editor, *CT-RSA 2006*, volume 3860 of *LNCS*, pages 226–243. Springer, Heidelberg, February 2006.

[BBPS19] Madalina Bolboceanu, Zvika Brakerski, Renen Perlman, and Devika Sharma. Order-lwe and the hardness of ring-lwe with entropic secrets. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, pages 91–120, Cham, 2019. Springer International Publishing.

[BD20] Zvika Brakerski and Nico Döttling. Lossiness and entropic hardness for ring-lwe. In *Theory of Cryptography Conference*, pages 1–27. Springer, 2020.

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Heidelberg, August 2018.

[BGM+16] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In *Theory of Cryptography Conference*, pages 209–224. Springer, 2016.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In *Theory of Cryptography: 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, pages 645–677. Springer, 2017.

[BJMS20] Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. Secure MPC: Laziness leads to GOD. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 120–150. Springer, Heidelberg, December 2020.

[BLRL+18] Shi Bai, Tancrède Lepoint, Adeline Roux-Langlois, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: using the rényi divergence rather than the statistical distance. *Journal of Cryptology*, 31(2):610–640, 2018.

[BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key fhe with short ciphertexts. In *Annual International Cryptology Conference*, pages 190–213. Springer, 2016.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference*, pages 868–886. Springer, 2012.

[BS23] Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from lwe with polynomial modulus. *Cryptology ePrint Archive*, 2023.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In Moni Naor, editor, *ITCS 2014*, pages 1–12. ACM, January 2014.

[CCK23] Jung Hee Cheon, Wonhee Cho, and Jiseung Kim. Improved universal thresholdizer from threshold fully homomorphic encryption. *Cryptology ePrint Archive*, 2023.

[CCS19] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Multi-key homomorphic encryption from tfhe. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 446–472. Springer, 2019.

[CDS15] Sergiu Carpov, Paul Dubrulle, and Renaud Sirdey. Armadillo: A compilation chain for privacy preserving applications. In *Proceedings of the 3rd International Workshop on Security in Cloud Computing*, SCC '15, page 13–19, New York, NY, USA, 2015. Association for Computing Machinery.

[ÇDSS15] Gizem S Çetin, Yarkın Doröz, Berk Sunar, and Erkay Savaş. Depth optimized efficient homomorphic sorting. In *International Conference on Cryptology and Information Security in Latin America*, pages 61–80. Springer, 2015.

[CGBH+18] Hao Chen, Ran Gilad-Bachrach, Kyoohyung Han, Zhicong Huang, Amir Jalali, Kim Laine, and Kristin Lauter. Logistic regression over encrypted data from fully homomorphic encryption. Cryptology ePrint Archive, Report 2018/462, 2018. `https://eprint.iacr.org/2018/462`.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security*, pages 409–437. Springer, 2017.

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 630–656. Springer, Heidelberg, August 2015.

[CMS+23] Sylvain Chatel, Christian Mouchet, Ali Utkan Sahin, Apostolos Pyrgelis, Carmela Troncoso, and Jean-Pierre Hubaux. Pelta–shielding multiparty-fhe against malicious adversaries. *Cryptology ePrint Archive*, 2023.

[CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 446–464. Springer, Heidelberg, April 2012.

[CO17] Wutichai Chongchitmate and Rafail Ostrovsky. Circuit-private multi-key FHE. In *PKC 2017*, volume 10175, pages 241–270, 2017.

[CS20] Ayantika Chatterjee and Indranil Sengupta. Sorting of fully homomorphic encrypted cloud data: Can partitioning be effective? *IEEE Transactions on Services Computing*, 13(3):545–558, 2020.

[CSS20] Gizem S Cetin, Erkay Savaş, and Berk Sunar. Homomorphic sorting with better scalability. *IEEE Transactions on Parallel and Distributed Systems*, 32(4):760–771, 2020.

[CZW17] Long Chen, Zhenfeng Zhang, and Xueqing Wang. Batched multi-hop multi-key FHE from ring-lwe with compact ciphertext extension. In *TCC 2017*, volume 10678, pages 597–627, 2017.

[DDFY94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *26th ACM STOC*, pages 522–533. ACM Press, May 1994.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer, Heidelberg, August 1990.

[DM15a] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640. Springer, Heidelberg, April 2015.

[DM15b] Léo Ducas and Daniele Micciancio. Fhew: bootstrapping homomorphic encryption in less than a second. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 617–640. Springer, 2015.

[DT06] Ivan Damgård and Rune Thorbek. Linear integer secret sharing and distributed exponentiation. In *International Workshop on Public Key Cryptography*, pages 75–90. Springer, 2006.

[DWF22] Xiaokang Dai, Wenyuan Wu, and Yong Feng. Key lifting: Multi-key fully homomorphic encryption in plain model without noise flooding. *Cryptology ePrint Archive*, 2022.

[Fra90] Yair Frankel. A practical protocol for large group oriented networks. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *EUROCRYPT'89*, volume 434 of *LNCS*, pages 56–61. Springer, Heidelberg, April 1990.

[FSK+21] Axel Feldmann, Nikola Samardzic, Aleksandar Krastev, Srini Devadas, Ron Dreslinski, Karim Eldefrawy, Nicholas Genise, Christopher Peikert, and Daniel Sanchez. F1: A Fast and Programmable Accelerator for Fully Homomorphic Encryption (Extended Version). *arXiv preprint arXiv:2109.05371*, 2021.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, 2012.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[GHL22] Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 458–487. Springer, 2022.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P Smart. Fully homomorphic encryption with polylog overhead. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 465–482. Springer, 2012.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.

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

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

[Hay08] Brian Hayes. Cloud computing, 2008.

[JRS17] Aayush Jain, Peter M. R. Rasmussen, and Amit Sahai. Threshold fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, page 257, 2017.

[KHF+19] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *2019 IEEE Symposium on Security and Privacy*, pages 1–19. IEEE Computer Society Press, May 2019.

[KS23] Kamil Kluczniak and Giacomo Santato. On circuit private, multikey and threshold approximate homomorphic encryption. *Cryptology ePrint Archive*, 2023.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234, 2012.

[LM21] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part I 40*, pages 648–677. Springer, 2021.

[LMK+22] Yongwoo Lee, Daniele Micciancio, Andrey Kim, Rakyong Choi, Maxim Deryabin, Jieun Eom, and Donghoon Yoo. Efficient fhew bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. Cryptology ePrint Archive, Paper 2022/198, 2022. `https://eprint.iacr.org/2022/198`.

[LMSS22] Baiyu Li, Daniele Micciancio, Mark Schultz, and Jessica Sorrell. Securing approximate homomorphic encryption using differential privacy. In *Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part I*, pages 560–589. Springer, 2022.

[LSG+18] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading kernel memory from user space. In William Enck and Adrienne Porter Felt, editors, *USENIX Security 2018*, pages 973–990. USENIX Association, August 2018.

[MBH22] Christian Mouchet, Elliott Bertrand, and Jean-Pierre Hubaux. An efficient threshold access-structure for rlwe-based multiparty homomorphic encryption. *Cryptology ePrint Archive*, 2022.

[MBTPH20] Christian Mouchet, Jean-Philippe Bossuat, Juan Troncoso-Pastoriza, and J Hubaux. Lattigo: A multiparty homomorphic encryption library in go. In *WAHC 2020–8th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, 2020.

[Mic18] Daniele Micciancio. On the hardness of learning with errors with binary secrets. *Theory of Computing*, 14(1):1–17, 2018.

[MS+11] Steven Myers, Mona Sergi, et al. Threshold fully homomorphic encryption and secure computation. *Cryptology ePrint Archive*, 2011.

[MTBH21] Christian Mouchet, Juan Ramón Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. Multiparty homomorphic encryption from ring-learning-with-errors. *Proc. Priv. Enhancing Technol.*, 2021(4):291–311, 2021.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Heidelberg, May 2016.

[PS16] Chris Peikert and Sina Shiehian. Multi-key fhe from lwe, revisited. In *Theory of Cryptography Conference*, pages 217–238. Springer, 2016.

[PV21] Alex Padron and Guillermo Vargas. Multiparty homomorphic encryption. *Online: https://courses. csail. mit. edu/6.857/2016/files/17. pdf*, 2021.

[RC19] B. Reddy and Ayantika Chatterjee. *Encrypted Classification Using Secure K-Nearest Neighbour Computation*, pages 176–194. 11 2019.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[Rot11] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In *Theory of cryptography conference*, pages 219–234. Springer, 2011.

[SCK14] Hassan Shee, Wilson Cheruiyot, and Stephen Kimani. Application of k-nearest neighbour classification in medical data mining. 4, 04 2014.

[SG02] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15(2):75–96, March 2002.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[SS10] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 377–394. Springer, Heidelberg, December 2010.

[TT15] Katsuyuki Takashima and Atsushi Takayasu. Tighter security for efficient lattice cryptography via the rényi divergence of optimized orders. In *International Conference on Provable Security*, pages 412–431. Springer, 2015.

[WVLY+10] Lizhe Wang, Gregor Von Laszewski, Andrew Younge, Xi He, Marcel Kunze, Jie Tao, and Cheng Fu. Cloud computing: a perspective study. *New generation computing*, 28(2):137–146, 2010.

A Observing the Pattern of Secret Shares

We state our observation on the pattern of the secret shares, generated by the (t, T)-threshold secret sharing using Benaloh-Leichter LISSS (Section 4.3), in the form of a theorem and provide the corresponding proof here.

Theorem 1. *$\mathcal{P}' = \{P_{id_1}, P_{id_2}, \dots, P_{id_t}\} \subset \mathcal{P} = \{P_1, P_2, \dots, P_T\}$ is a t-sized group with* group_id *value of gid, where $id_1 < id_2 < \cdots < id_t$. $\forall 1 \leq i \leq t$, P_{id_i} has a key share SH_i, tagged with* group_id *value of gid. Then all key shares except SH_1, have only binary coefficients in their k polynomials, while SH_1 will have coefficient value upper-bounded by t in its k polynomials.*

In order to prove Theorem 1, we will first state two lemmas related to the structure of the distribution matrix M for (t, T) threshold secret sharing of a TRLWE secret key S. We consider the number of polynomials in S is k and I_k denotes the identity matrix of dimension k.
The first lemma is about the pattern of the distribution matrix for Boolean formula of the form $x_1 \wedge x_2 \wedge \cdots \wedge x_t$ for any t.

Lemma 1. *We consider $\mathbf{0}$ to be a notation of zero matrix of dimension $k \times k$. Then, distribution matrix M_f for Boolean formula $f = x_1 \wedge x_2 \wedge \cdots \wedge x_t$ follows the following structure.*

$$\begin{bmatrix} I_k & I_k & I_k & \dots & I_k & I_k \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & I_k \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & I_k & \mathbf{0} \\ \vdots & & & & & \\ \mathbf{0} & \mathbf{0} & I_k & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & I_k & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}_{kt \times kt}$$

Proof of Lemma 1. We prove the lemma by induction on the value of t.
For $t = 1$, $f = x_1$ and $M_f = I_k$. Hence, the stated matrix structure is satisfied by default.
For $t = 2$, $f = x_1 \wedge x_2$. We follow the ANDing procedure (see Section 4.3) of $M_{x_1} = I_k$ and $M_{x_2} = I_k$ and get $M_{x_1 \wedge x_2} = \begin{bmatrix} I_k & I_k \\ \mathbf{0} & I_k \end{bmatrix}$, which clearly satisfies the claimed structure.
Let us assume that the claimed structure of the distribution matrix holds for $t = i$, i.e., for $f = x_1 \wedge x_2 \wedge \cdots \wedge x_i$, M_f is as shown below. Also, x_{i+1} being a Boolean variable, $M_{x_{i+1}} = I_k$. ANDing M_f and $M_{x_{i+1}}$ produces $M_{f_1} = M_{f \wedge x_{i+1}}$ as shown below. M_f has a dimension of $ki \times ki$ and M_{f_1} has a dimension of $k(i+1) \times k(i+1)$.

$$M_f = \begin{bmatrix} I_k & I_k & I_k & \dots & I_k & I_k \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & I_k \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & I_k & \mathbf{0} \\ \vdots & & & & & \\ \mathbf{0} & \mathbf{0} & I_k & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & I_k & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}$$

$$M_{f_1} = \begin{bmatrix} I_k & I_k & I_k & I_k & \dots & I_k & I_k \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & I_k \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & I_k & \mathbf{0} \\ \vdots & & & & & & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I_k & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I_k & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & I_k & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}$$

Clearly, the structure is maintained for $t = i + 1$. Hence, by induction, the lemma is true for any $t \geq 1$. □

And the second lemma is about the pattern of distribution matrix for Boolean formula consisting of disjunction of l number of such t-sized conjunctive terms, i.e., $(x_{1,1} \wedge x_{1,2} \wedge \dots \wedge x_{1,t}) \vee \dots \vee (x_{l,1} \wedge x_{l,2} \wedge \dots \wedge x_{l,t})$.

Lemma 2. *Let us assume that $f' = (x_{1,1} \wedge x_{1,2} \wedge \dots \wedge x_{1,t}) \vee \dots \vee (x_{l,1} \wedge x_{l,2} \wedge \dots \wedge x_{l,t})$ is a Boolean formula, where $\forall 1 \leq i \leq l, 1 \leq j \leq t$, $x_{i,j}$ is a binary variable and each of the $(x_{i,1} \wedge x_{i,2} \wedge \dots \wedge x_{i,t})$ terms is represented by distribution matrix M_f, as stated in Lemma 1. We denote first k columns of M_f by F of dimension $kt \times k$ and the rest of the columns of M_f by R of dimension $kt \times k(t-1)$. $\mathbf{0}$ denotes zero matrix of dimension $kt \times k(t-1)$. Then distribution matrix $M_{f'}$ has the following structure:*

$$\begin{bmatrix} F & R & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ F & \mathbf{0} & R & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & & & & & \\ F & \mathbf{0} & \dots & \mathbf{0} & R & \mathbf{0} \\ F & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & R \end{bmatrix}_{lkt \times (lkt-(l-1)k)}$$

Proof of Lemma 2. We prove the lemma by induction on the value of l.
For $l = 1$, $f' = f = (x_{1,1} \wedge x_{1,2} \wedge \dots \wedge x_{1,t})$ and $M_{f'} = M_f = \begin{bmatrix} F & R \end{bmatrix}$, which satisfies the claimed structure by default.
For, $l = 2$, $f' = (x_{1,1} \wedge x_{1,2} \wedge \dots \wedge x_{1,t}) \vee (x_{2,1} \wedge x_{2,2} \wedge \dots \wedge x_{2,t})$. We perform ORing on $M_{x_{1,1} \wedge x_{1,2} \wedge \dots \wedge x_{1,t}} = M_f$ and $M_{x_{2,1} \wedge x_{2,2} \wedge \dots \wedge x_{2,t}} = M_f$ (see Section 4.3) and get

$$M_{f'} = \begin{bmatrix} F & R & \mathbf{0} \\ F & \mathbf{0} & R \end{bmatrix}_{2kt \times (2kt-k)}$$

This structure follows the lemma.
Let us assume that the structure is maintained $\forall l \leq j$. So, with $f' = (x_{1,1} \wedge x_{1,2} \wedge \dots \wedge x_{1,t}) \vee \dots \vee (x_{j,1} \wedge x_{j,2} \wedge \dots \wedge x_{j,t})$ and $f'' = (x_{j+1,1} \wedge x_{j+1,2} \wedge \dots \wedge x_{j+1,t})$, $M_{f'}$ has a dimension of $jkt \times jkt - (j-1)k$ and $M_{f''}$ has a dimension of $kt \times kt$. $M_{f'}$ follows the structure as shown below. $M_{f''} = \begin{bmatrix} F & R \end{bmatrix}$. Now, ORing $M_{f'}$ and $M_{f''}$ produces $M_{f_2} = M_{f' \vee f''}$ with dimension $(j+1)kt \times ((j+1)kt - jk)$ as shown below.

$$M_{f'} = \begin{bmatrix} F & R & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ F & \mathbf{0} & R & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & & & & & \\ F & \mathbf{0} & \dots & \mathbf{0} & R & \mathbf{0} \\ F & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & R \end{bmatrix}$$

$$M_{f_2} = \begin{bmatrix} F & R & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ F & \mathbf{0} & R & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots \\ F & \mathbf{0} & \dots & \mathbf{0} & R & \mathbf{0} & \mathbf{0} \\ F & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & R & \mathbf{0} \\ F & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & R \end{bmatrix}$$

So, the lemma is true for $l = (j + 1)$.
Hence, by induction the lemma is true for any $l \geq 1$. □

Now we use Lemma 1 and Lemma 2 to provide here the proof of Theorem 1.

Proof of Theorem 1. Let us recall from Section 4.3 that the monotone Boolean formula for (t, T)-threshold secret sharing can be written as $f = (x_{1,1} \wedge x_{1,2} \wedge \dots \wedge x_{1,t}) \vee \dots \vee (x_{l,1} \wedge x_{l,2} \wedge \dots \wedge x_{l,t})$, where $l = \binom{T}{t}$. If $\mathbf{0}$ denotes zero matrix of dimension $kt \times (kt - k)$, from Lemma 1 and Lemma 2, we know that structure of the corresponding distribution matrix M with dimension $\binom{T}{t}kt \times (\binom{T}{t}kt - (\binom{T}{t} - 1)k)$ is as follows:

$$M = \begin{bmatrix} F & R & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ F & \mathbf{0} & R & \mathbf{0} & \dots & \mathbf{0} \\ \vdots \\ F & \mathbf{0} & \dots & \mathbf{0} & R & \mathbf{0} \\ F & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & R \end{bmatrix}$$

$$F = \begin{bmatrix} I_k \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \qquad R = \begin{bmatrix} I_k & I_k & I_k & \dots & I_k \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & I_k \\ \vdots \\ \mathbf{0} & I_k & \mathbf{0} & \dots & \mathbf{0} \\ I_k & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}$$

A detailed look into the above matrix M reveals that F has a structure of dimension $kt \times k$ and R has a structure with dimension $kt \times (kt - k)$ as shown in above matrix structure. In F and R, $\mathbf{0}$ denotes a zero matrix of dimension $k \times k$. It is obvious from the structure of M that each of its $\binom{T}{t}$ horizontal sections contain exactly one F and one R along with $(\binom{T}{t} - 1)$ zero matrices $\mathbf{0}_{kt \times (kt-k)}$. Now, the structure of F shows that each of its first k rows contains one '1' entry. No other row below has any '1' in it and the structure of R reveals that each of its first k rows contains exactly $(t - 1)$ number of '1' in it. Each of the other rows below contains exactly one '1' in it. Hence, each of the first k rows of any one horizontal section (out of total $\binom{T}{t}$ sections) of M has exactly t number of '1' in it. Each of the rest of the rows below in that section contains exactly one '1' in it.
Let us recall that, each section of M corresponds to one section of *shares* ($shares = M \cdot \rho$) from Section 4.3 in the paper, i.e, the key shares of any t-sized subset of collaborating parties.
ρ is a binary matrix. During matrix multiplication, dot product between one row of M and one column of ρ produces an entry in *shares*. Dot product between two binary vectors is always upper bounded by the number of '1' in any of the two vectors. As, each of first k rows of any section of M contains exactly t number of '1', the entries of first k rows of any

section in *shares* are always upper bounded by t. First k rows of any section of *shares* form one key-share. Clearly, that key share will have non-binary entries in it. Similarly, each of the other $(kt - k)$ rows below in any section of M contains exactly one '1', so the entries of the $(kt - k)$ number of rows below in any section of *shares* are upper bounded by 1. In other words, those entries can be either 0 or 1. Hence, rest of the $(t - 1)$ key shares of any t-sized subset of parties, have only binary entries in it.

Hence we conclude that, in our proposed (t, T) threshold LISSS for a t-sized subset of parties $PT' = \{P_{id_1}, P_{id_2}, \ldots, P_{id_t}\}$, where $id_1 < id_2 < \cdots < id_t$, all the parties except P_{id_1} will have binary key shares. $\square$