Threshold FHE with Efficient Asynchronous Decryption

Abstract. A *Threshold Fully Homomorphic Encryption (`ThFHE`)* scheme enables the generation of a global public key and secret key shares for multiple parties, allowing any threshold of these parties to collaboratively decrypt a ciphertext without revealing their individual secret keys. By leveraging the homomorphic properties of FHE, this scheme supports the distributed computation of arbitrary functions across multiple parties. As distributed execution of cryptographic tasks becomes popular, the demand for `ThFHE` schemes grows accordingly. We identify three major challenges with existing solutions. (i) They often take unrealistic assumptions with regards to the network model, assuming the threshold of parties to participate in decryption is known a-priori, available throughout multiple communication rounds, and is consistent between parties. (ii) They incur a super-linear overhead on the underlying FHE public parameters. Both issues pose challenges on scaling with the number of parties. (iii) The require heavyweight *Zero-Knowledge Proofs (ZKPs)* during decryption, thereby introducing a significant computational overhead in order to tolerate malicious behavior.

In this work, we introduce a `ThFHE` scheme that faces the above three challenges, and is designed to scale with the number of parties N.

Our scheme operates within the well-established asynchronous communication model. At the same time, upon decryption, the ciphertext only incurs a linear $\frac{3}{4}N + t$ additive overhead on the ciphertext modulus size. Additionally, when allowed to rely on *non Post Quantum (PQ)-secure* additively homomorphic encryption schemes, we provide a method with an $O(1)$ overhead, independent of N. Lastly, we propose a *preprocessing* technique, that allows the parties to *batch* and preprocess all necessary ZKPs in an offline phase, before the encrypted inputs and evaluation circuit are determined. In turn, this enables the system to effectively manage traffic spikes, by exploiting idle periods to preform the ZKPs.

We build on a ring-based FHE scheme, specifically using the BGV scheme for clarity and concreteness. Nonetheless, the techniques also apply to BFV, CKKS, and TFHE schemes.

1 Introduction

Fully Homomorphic Encryption (FHE) [RAD78, Gen09a], which enables the computation of arbitrary functions directly on encrypted data without requiring decryption, has seen widespread use in privacy-preserving computation and other cryptographic applications. These include secure authentication [CDC24], federated learning [HAG+23], CNN inference [KG23] among others. Notably,

FHE has proven particularly useful in the design of secure multiparty computation protocols (MPC), where parties jointly compute a functionality over their private inputs [DPSZ12, LATV12, LPSY19, OSV20]. A popular approach in the use of FHE for MPC is to follow this template [Gen09b]:

1. *Distributed Key Generation (DKG).* The parties collaboratively compute a single FHE public key pk, with the corresponding secret key sk being secret-shared among them, such that each party P_i holds a share sk_i.
2. *Broadcast Encrypted Inputs.* Each party encrypts their private input individually using the public key pk, resulting in the input of party P_i, denoted as pt_i, being encrypted into a ciphertext ct_i.
3. *Homomorphic Evaluation.* Each party, individually, homomorphically evaluates the input ciphertexts ct_i, obtaining an encryption of the output $\mathsf{ct}_{\mathsf{out}}$.
4. *Threshold Decryption.* The parties jointly decrypt $\mathsf{ct}_{\mathsf{out}}$ using their individual shares sk_i to obtain the output of the desired circuit.

Notably, in some settings, parties who send encrypted inputs may not hold any shares of the private key. Similarly, since the homomorphic evaluation step involves public computation, a separate network may be responsible for this step as well. In this work, we mainly focus on boosting threshold decryption.

The above high-level template has been instantiated in many variants. One of its most appealing properties is the ability to decouple the complexity of the MPC protocol from the complexity of the functionality to be computed. The functionality only affects the homomorphic evaluation, which occurs offline, and potentially even off-loaded to another distributed network. Another advantage is that the initial and final steps are agnostic to the functionality and can be performed by parties who do not need to be aware of the specifics of the functionality, as long as they agree to participate in the DKG and decryption processes.

The aforementioned template was utilized by Asharov et al. [AJLA+12] to reduce the round complexity of MPC. Subsequently, it was extended in the context of multi-key FHE [LATV12, CM15, MW16], a primitive that enables completely non-interactive key generation (in a semi-honest setting).

In previous solutions, all parties involved in the key generation were also required to participate in the decryption process. However, Boneh et al. [BGG+18] proposed deviating from this requirement by introducing *threshold* fully homomorphic encryption (`ThFHE`). In `ThFHE` schemes, decryption can proceed as long as a sufficiently large subset of honest parties—defined by a threshold parameter t—comes together. They also demonstrated that `ThFHE` can be used to construct a powerful primitive they term a "universal thresholdizer," which can incorporate threshold functionality into virtually any cryptographic functionality.

The focus of `ThFHE` is on the concept of *distributed decryption* with minimal interaction. Specifically, it requires that each decrypting party sends only a single message. By collecting messages from more than t honest parties, it should be possible to recover the encrypted message.

Ideally, the system should work under an asynchronous communication channel. In the context of threshold decryption, this introduces challenges. First, parties computing their individual decryption messages, termed *decryption shares*,

may not be aware of the availability of other parties or which specific set of $t+1$ decryption shares will be used for decryption. Second, different parties may collect a different subset of $t+1$ decryption shares, as messages can arrive out of order.[1] We call this the *asynchronous decryption* setting, where parties need not synchronize for successful decryption. Implementing such synchronization would significantly increase complexity and reduce robustness.

In order to illustrate the challenge with thresholdizing lattice-based schemes, we consider below a standard Learning-With-Errors (LWE) BGV scheme. However, a similar discussion can be provided for other FHE schemes such as TFHE, and BFV, CKKS, also in their ring (RLWE) and module (MLWE) variants. For ease of exposition, consider encrypting a plaintext $m \in \mathbb{Z}_p$, using an LWE BGV ciphertext of the form $(\boldsymbol{a}, b)$ with ciphertext modulus q, where $\boldsymbol{a} \in \mathbb{Z}_q^n$, $b \in \mathbb{Z}_q$, using the equation $b = \boldsymbol{a} \cdot \boldsymbol{s} + m + p \cdot e \mod q$, where $e \leftarrow \chi_{\mathsf{E}}$ is some "noise" term and $\boldsymbol{s} \in \mathbb{Z}_q^n$ is the secret key. To support any threshold parameter t, known solutions, starting with [BGG+18], adapt Shamir-style threshold secret sharing of the secret FHE key $\boldsymbol{s}$, denoted $[\boldsymbol{s}] = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N)$. One can trivially produce a secret sharing of the value $m + p \cdot e$ by computing $[\mathsf{ds}] = b - \boldsymbol{a} \cdot [\boldsymbol{s}] = [m + p \cdot e]$. Indeed, by opening the value $[\mathsf{ds}]$ all parties can then take modulo p to obtain m. However, introduces two issues. First, this procedure also reveals the value of e, which combined with the ciphertext and the message, exposes information on the secret key $\boldsymbol{s}$. Additionally, each decryption share $\mathsf{ds}_i = b - \boldsymbol{a} \cdot \boldsymbol{s}_i$, gives away a linear equation on the secret key share of the corresponding party. Therefore, after n threshold decryptions, all secret shares are trivially reconstructed.

Notably, some works attempted to mitigate the challenge of thresholdizing lattice-based schemes by restricting the access structure. For example, [CLO+13, BGG+18], and a variant by [DDEK+23] wherein $\binom{N}{t}$ is bounded. However, this compromises security, as fewer parties increase the risk of silent collusion. This diverges from our focus on scaling with the threshold and number of parties, aiming for distributed, collusion-resistant solutions. In [ABGS23], the secret key is additively shared, requiring all parties in decryption. Unfortunately, this is not robust for large-scale networks due to potential failures.

One approach to resolve the two issues above, taken by [DDEK+23], is to add a pre-processing phase in which the parties derive an additional secret shared noise term $[E]$ that statistically hides the error. This technique is known as *noise flooding*. Thus, the parties may now open $[\mathsf{ds}] = b - \boldsymbol{a} \cdot [\boldsymbol{s}] + p \cdot [E] = [m + p \cdot (e + E)]$. Unfortunately, generation of a secret share requires Peer-to-Peer channels, in which a threshold of parties send unique secret shares to each party privately. In particular, each party computes, receives and processes N messages, and so this approach does not scale well with the number of parties.

In this work, we follow a different approach originally proposed in [BGG+18]. There, each decrypting party adds a locally sampled error to its decryption share: $\mathsf{ds}_i = b - \boldsymbol{a} \cdot \boldsymbol{s}_i + p \cdot E_i$, with $E_i \leftarrow \hat{\chi}_{\mathsf{E}}$ sampled independently. While this prevents exposing equations on the secret key shares, reconstructing the plaintext becomes

[1] This can be resolved by implementing a consensus mechanism, which will further increase network latency.

unclear. Upon reconstruction, decryption shares are multiplied by corresponding Lagrange coefficients, which can be arbitrarily large. As a result, parties obtain $m + p \cdot e + p \cdot \sum_i \lambda_i \cdot E_i \mod q$, from which m cannot be reliably extracted.

To resolve this issue, [BGG+18] shows that by "clearing the denominator" of the Lagrange coefficients, which essentially means multiplying all coefficients by the least common multiple (LCM) of their denominators (see [Sho00, ABV+12]), it is possible to bound the noise increase by a factor of $(N!)^2$. To manage this noise increase, the ciphertext modulus must be increased accordingly, approximately by $\mathcal{O}(N \log N)$ bits. This comes in addition to $\sigma/2$ bit increment from noise flooding, where σ is the statistical security parameter. In turn, the ring and module degrees should be adjusted, in order to maintain security. More of the same, the above penalty prevents scaling by the number of parties N.

Alternatively, to bypass the complications arising from large Lagrange coefficients, some works such as [MTPBH21, MBH23] abandon the asynchronous model altogether. Indeed, if the threshold of parties to be sending decryption shares is known in advance, the Lagrange coefficients can be precomputed, and so $\mathsf{ds}_i = b - \boldsymbol{a} \cdot \lambda_i \boldsymbol{s}_i + p \cdot E_i$ becomes a t-out-of-t additive sharing on $m + p \cdot e + p \sum_i E_i$. However, as mentioned, the synchronous model compromises on the latency of the system, and is challenging to realize and maintain.

1.1 Our Contribution

Given the significance of threshold functionalities in modern cryptography, we investigate the following question:

Is it possible to develop a ***ThFHE*** *scheme under an asynchronous broadcast communication channel, that scales with respect to the number of parties?*

In this work, we answer in the affirmative. Since DKG is one-time, in this work we are focused on boosting threshold decryption. For completeness, we offer a potential instantiation of DKG in Appendix C, implemented over a consensus channel. In particular, while messages are received in the same order for all parties, the subset of participating parties may arbitrarily change in each communication round and is post-determined. This poses a few challenges that we discuss, both for correctness and UC security.

Below, we break down our optimizations for threshold decryption into four milestones. The corresponding techniques are introduced in Section 1.3. A comparison with prior schemes is provided in Table 1.

Our first result demonstrates that in RLWE based FHE schemes, it is possible for the ciphertext moduli to grow only linearly by the number of parties $\mathcal{O}(N)$. Concretely for $N = 360$, the method in [BGG+18], would incur an increse of 5086 bits to the ciphertext modulus while our scheme will incur a 593 bits increase. This also has implications on the needed ring degree for security. In short while still a large overhead this improvement makes the denominator clearing approach feasible for hundreds of parties.

Our second result is that by utilizing key-switching, it is possible to decouple the parameters for threshold decryption from those of the homomorphic evaluation, further minimizing the overhead of the threshold decryption phase. This

means that complex homomorphic operations are carried over with parameters of the typical (non-threshold) FHE underlying scheme. In particular, the number of parties does not affect the complexity of bootstrapping.

Our third result demonstrates that by adding a pre-processing round, it is possible to alleviate the need for the verification of Zero-Knowledge Proofs (ZKPs) of correctness of the decryption shares. In case the adversary inputs a false decryption share, failure is detected, and only in this case proof verification may be used to detect malicious parties. Intuitively, the idea is that instead of recovering the function output, parties derive the encryption noise. This not only allows recovery of the underlying plaintext, but also allows parties to verify its validity. Additionally, this also suggests that a single party may be responsible for combining the decryption shares and deriving the encryption noise, which can be used as a witness of correct decryption.[2] The latter improves the *amortized decryption* computational cost by N. In addition, since the pre-processing round is independent of the inputs and function to be evaluated, the ZKPs of this round can be batched, which in turn significantly reduces the overhead of handling malicious behavior. The output of this pre-processing phase may then be used for multiple, potentially parallel, online threshold decryption rounds.

Lastly, we show that when relying on non-Post Quantum (PQ) assumptions[3], it is possible to reach an $\mathcal{O}(1)$ additive overhead. This is done by utilizing a threshold additively homomorphic encryption scheme with such overhead for decryption. For instance, Tiresias [FMM+23] based on Paillier [Pai99], or [BDO23, BCD+24] based on the Class-Groups encryption scheme by [CLT18] in the CL framework [CL15].

1.2 Comparison with Prior Work

A comparison of the properties of our work with some previously mentioned works is presented in Table 1 below.

In our native RLWE protocol, we follow a similar approach to the work in [BGG+18], of multiplying the noise terms by a denominator clearing factor Δ, in order to ensure correctness of decryption. As mentioned, we improve upon their choice of Δ, which results with an asymptotic improvement in the ciphertext modulus, and in turn the decryption share size as well.

In comparison, the approach taken by [BS23] is to let the decryptors enumerate over all possible choices of the subset of parties to participate in the online phase. As a result, the noise terms additively scale with N. However, each decryptor sends $O(\binom{N}{t})$ decryption shares, which does not scale well (N, t). We note [DDEK+23] also describe a variant threshold decryption for the case where

[2] In fact, this can be used to prove correctness of decryption toward the client, who may not have the computational and communicational capacity to verify the decryption shares of all parties in the network. Nonetheless, it would still have to rely on the entities responsible for the homomorphic computation itself.

[3] That is, based on problems efficiently solvable by quantum computers (BQP), e.g. factoring.

$\binom{N}{t}$ is small. This affects the computational complexity of the online decryption phase, yet the decryption shares remain compact. However, in this work we focus on scaling with N and therefore compare with their second variant. The size of decryption shares was improved in [CCK23] to scale with $\tilde{O}(N^{4.3})$. Despite the improved result being polynomial with N, in practice for hundreds to thousands of parties this factor is quite large.

As mentioned, a key challenge with thresholdizing FHE schemes is to achieve asynchronous decryption. That is, during the online threshold decryption phase, each party that sends its decryption share does not know in advance the set of participants that will be online. In [MBH23], it is shown that if the set of decryptors is pre-determined, the decryption share size may grow logarithmically with N. Notably, this work also proposes a DKG protocol for an RLWE-based encryption scheme, that generates the public key and secret decryption shares, which only requires an Asynchronous Reliable Broadcast (ARB) channel. However, their protocol is provided in the semi-honest model, and heavyweight ZKPs may be needed to make secure against malicious adversaries. In addition, their DKG does not specify how to generate the homomorphic evaluation keys.

We also mention [ABGS23], who utilize a lattice-based threshold encryption scheme for digital voting. Their scheme uses additive N-out-of-N sharing, so the participant set is fixed and decryption shares are compact. Nevertheless, as $t = N - 1$, the system cannot scale with N from a more practical perspective, as it is not resilient to dropouts. On the other hand, they provide a scheme that is secure against malicious adversary, by using proper ZKPs that can be efficiently batched for multiple statements, which we adopt in this work.

Lastly, we discuss the scheme proposed in [DDEK+23] in the context of large N and t. Their scheme incorporates a DKG protocol that generates not only the public key and decryption shares but also the bootstrapping key, enabling a threshold fully homomorphic encryption (TFHE) scheme. Furthermore, their protocol is secure against malicious adversaries and achieves *robustness*, avoiding zero-knowledge proofs (ZKPs). Instead, they employ Reed-Solomon error correction over a Galois ring. However, their approach has a key limitation: the decryption threshold is restricted to $t < N/3$ for correctness. This poses a significant practical issue. While our protocol also requires $t < N/3$, this constraint arises solely from broadcast assumptions. In practice, attacks on the broadcast channel necessitate network forking, making them detectable and difficult to execute. Another drawback of their scheme lies in the offline phase, where parties generate a sharing of a fresh secret $[E]$. This step relies on a robust multiplication protocol from [ACD+19], which enables threshold secret sharing over binary secrets, allowing parties to locally derive a sharing of $[E]$. However, this approach does not scale efficiently with N. It entails numerous communication rounds, where parties secret-share elements, jointly sample challenges, and reconstruct values to verify correctness, leading to significant overhead.

In this work, we also utilize preprocessing techniques in order to enhance the performance during the online threshold decryption phase. However, our offline phase consists of a single broadcast round where parties send a batch of cipher-

text along with a batched ZKP. This makes the offline phase both intuitive and efficient, yet allows the online phase to not involve heavyweight ZKPs. Our work also provides a DKG protocol based only on an asynchronous broadcast consensus channel. It generates the public FHE encryption key, the secret decryption shares, as well as all the necessary public evaluation key parameters.

Scheme	Async Dec.	Modulus Increase	Dec. Share Size	Setup	Online ZKPs	Offline Phase
[BGG+18]	✓	$O(N\log(N))$	$O(N\log(N))$	Trusted	SH	✗
[BS23]	✓	$O(\log(N))$	$O(\log(N)\binom{N}{t})$	Trusted	SH	✗
[CCK23]	✓	$O(\log(N))$	$O(\log(N)N^{4.3})$	Trusted	SH	✗
[MBH23]	✗	$O(\log(N))$	$O(\log(N))$	ARB	SH	✗
[DDEK+23]	✓	$O(1)$	$O(1)$	P2P	Light	Heavy
Ours (PQ)	✓	$O(N)$	$O(N)$	ARB	Light	Light
Ours	✓	$O(1)$	$O(1)$	ARB	Light	Light

Table 1: **Comparison to Existing `ThFHE` Schemes**. N is the number of parties, t is threshold. Our protocols refer to the version presented in Section 6 Protocol 2, that leverages preprocessing techniques, depending on whether the construction plausibly maintains Post Quantum (PQ) security. In the setup column, schemes that rely on a trusted setup are marked 'trusted'. Otherwise, P2P and ARB refer to the underlying communication channel for implementing the scheme. In this context, P2P stands for secure Peer-to-Peer channels while ARB stands for Asynchronous Reliable Broadcast. Specifically, P2P channels are used in [DDEK+23] during the offline preprocessing phase. In all schemes, the decryption shares are broadcast. Schemes in the semi-honest (SH) setting require generic heavywight ZKPs in order to handle malicious adversaries. Schemes are considered to have lightweight ZKPs, when those can be efficiently batched and offloaded to a preprocessing phase. In particular, the online threshold decryption phase does not involve verification of decryption shares. Finally, we mark which scheme use a pre-processing phase. The preprocessing in [DDEK+23] is considered heavyweight, as it involves robust multiplication over P2P channels, in order to secretly share each bit of the decryption error, using [ACD+19]. In comparison, our pre-processing is a single broadcast round, with batched ZKPs, which scales more efficiently with N.

1.3 Technical Overview

Achieving asynchronous threshold encryption schemes for schemes not based on lattices (e.g., Tiresias [FMM+23] based on Paillier [DJN10], Discrete Logarithm [DF91], or Class Groups [BDO23]) is typically straightforward. This is because the linear relations directly carry over. However, this is not the case for lattice-based schemes. As mentioned earlier, in the lattice world, the presence of encryption noise poses some challenges.

One approach to address this issue, which we adopt and enhance, is "denominator clearing" [Sho00, ABV+12, BGG+18]. This technique involves treating the error as a multiple of a specific value Δ which effectively removes all denominators in the reconstruction coefficients. Following our running example, the ciphertext now admits $b = \boldsymbol{a} \cdot \boldsymbol{s} + m + \Delta \cdot p \cdot e$, the decryption share error term is also multiplied by Δ, and so $\mathsf{ds}_i = b - \boldsymbol{a} \cdot \boldsymbol{s}_i + \Delta \cdot p \cdot E$. Then, upon reconstruction, the parties retrieve $m + p(\Delta e + \sum_i \Delta \lambda_i E_i) \mod q$. Importantly, $\Delta\lambda_i$ are now all integral, and so if q is large enough, the plaintext m can be extracted. For this reason, it is best to have Δ as small as possible, which will translate to tighter public parameters for the threshold encryption scheme, and improve performance. However, inevitably, Δ does grow with the number of parties N.

The state-of-the-art approach [BGG+18] for arbitrary threshold yields $\Delta = (N!)^2$, leading to an increase in the bit-length of the ciphertext modulus by approximately $\log \Delta = \mathcal{O}(N \log N)$. We observe that the high cost associated with the above method is mainly due to the dependence on the assumption of LWE (Learning with Errors). This is because the ring $\mathbb{Z}$ lacks a variety of small elements. In contrast, multidimensional rings, such as polynomial rings, contain many elements with small norms—often including at least one element of norm one for each party in relevant scenarios (e.g., considered in this work). By selecting these small-norm elements as interpolation points for Shamir Secret Sharing, we can achieve a solution where the magnitude is $\Delta \leq 2^{2N}$. With a more detailed analysis, using symmetry and fundamental algebraic properties, we can further reduce this factor to $\approx 2^{\frac{3}{4}N}$.

Specifically, we select our interpolation points as $\pm x^i$ and set our denominator clearing factor to $\Delta = (x^2-1)\cdot(x^4-1)\cdots(x^{\frac{N}{2}}-1)\cdot(x^2-1)\cdots(x^{\frac{N}{6}}-1)$. At first glance, it may seem that there are not enough terms to cancel the denominator of the Lagrange coefficient; however, the identity $x^2 - 1 = (x+1)(x-1)$, along with some combinatorial analysis, demonstrates that it effectively clears the denominator. When assessing the overall impact on error size, we must bound terms of the form $(x-1)\cdot(x^2-1)\cdots(x^N-1)$. To achieve this, we adopt an elegant mathematical result from [Wri64]. We refer to Section 4 for details.

Notably, for at least a random subset of parties, we observe that the magnitudes are significantly smaller than what we can formally prove. In Section D, we provide both experimental and heuristic analyses based on mathematical conjectures regarding Sudler products, suggesting that the average factor is approximately $O(2^{0.12N})$. This may significantly improve parameter estimation; in particular, the scheme can be instantiated with realistic parameters for up to a thousand parties. Unfortunately, we were unable to prove this heuristic, and there are technical challenges preventing us from fully taking advantage of it.

To make our threshold scheme maliciously secure, we turn the above secret sharing scheme over the ring into a *publicly verifiable secret sharing (PVSS)* scheme by attaching each secret share sk_i of the secret key sk with a *public verification key* $\mathsf{vk}_i = \mathbf{com}(\mathsf{sk}_i)$, which can be thought of as a commitment to the secret key. Upon decryption, each party sends its decryption share ds_i along with a zero-knowledge proof $\Pi_i^{\mathsf{ds}}(\mathsf{ct}, \mathsf{ds}_i, \mathsf{vk}_i; \mathsf{sk}_i, \cdot)$ that proves its correctness.

Optimizations. The approach described above suffers from several factors: (i) the size of the input ciphertexts is depend on the number of parties N, which in turn affect the cost of homomorphic evaluation; (ii) the protocol is heavily built on ZKP, and each decryptor must prove their partial decryption and verify at least t proofs; and (iii) each party is required to perform their own recombination of shares, but, in certain applications, it might be more effective for one party to perform the computation. Below, we outline our approach to address these weaknesses (see Section 6 for details).

The first point (i) can be resolved by applying a key-switch right before threshold decryption. Namely, the input ciphertext and homomorphic evaluation phase carry over with parameters independent of N, resulting with a ciphertext ct such that $\mathsf{ct}_0 + \mathsf{ct}_1 \cdot \mathsf{sk} = \mathsf{pt} + p \cdot e \mod q$. We then apply a key-switch in order to get a ciphertext of the form $\mathsf{ct}'_0 + \mathsf{ct}'_1 \cdot \mathsf{sk} = \mathsf{pt} + \Delta \cdot p \cdot e \mod Q$, wherein Q now depends on N.

For (ii), our key idea is to derive an encryption of the randomizer u corresponding to the ciphertext ct'. Threshold decryption is then applied to obtain the randomizer u rather than the underlying plaintext pt of ct' directly. An honest party can use the randomizer u not only to derive the plaintext pt, but also to verify that ct' is well-formed. Specifically, let $(\mathsf{pk}_0, \mathsf{pk}_1)$ be the associated public key and assume $\mathsf{ct}'_0 = u\mathsf{pk}_0 + pe_0, \mathsf{ct}'_1 = u\mathsf{pk}_1 + \mathsf{pt} + pe_1$. Then an honest party can compute $pe_0 = \mathsf{ct}'_0 - u\mathsf{pk}_0$, and $\mathsf{pt} + pe_1 = \mathsf{ct}'_1 - u\mathsf{pk}_1$, extract e_0, e_1 and pt and check that the noises e_0, e_1 are within bounds. Nevertheless, the ciphertext ct' may result from a homomorphic evaluation rather than fresh encryption, and tracking the randomizer through operations such as modulus switching, key switching, or bootstrapping is tricky. We therefore propose computing the randomizer directly from ct'.

To do that, we leverage the fact that the ciphertext ct' for decryption is a result of a key-switching operation. Since key-switching involves a subset-sum of the key-switching keys ks, we can express the randomizer u of ct' as a known subset-sum of the randomizers of the key-switching keys u_{ks}. Mathematically, $u = u_{\mathsf{ks}}\mathsf{ct}'_1$, where $\mathsf{ct}' = (\mathsf{ct}'_0, \mathsf{ct}'_1)$ is the ciphertext before the key-switch. Therefore, during the DKG the parties also generate encryptions of u_{ks}, and only the parameters for those ciphertexts depend on N.

However, this approach incurs a security issue: computing the randomizer u in the clear exposes a linear equation on u_{ks}, which together with the key-switching keys may compromise the secret key. To mitigate this, we propose to statistically hide the randomizer u of the ciphertext ct' in an offline phase with noise flooding. Concretely, in an offline phase, $t+1$ parties broadcast encryptions of zeros $\widehat{\mathsf{ct}}^i = \mathsf{E}(0, \hat{u}_i)$ along with $\mathsf{E}(\hat{u}_i)$ and a zk-proof that ties the two. The existence of one honest party ensures that $\hat{u} = \sum_i \hat{u}_i$ statistically hides u[4], and the proofs ensure correctness of decryption. Then, instead of decrypting $\mathsf{E}(u) = \mathsf{E}(u_{\mathsf{ks}}) \cdot \mathsf{ct}'_1$, the parties decrypt $\mathsf{E}(u+\hat{u})$, and validate it against $\mathsf{ct}' + \sum_i \widehat{\mathsf{ct}}^i$.

[4] A rushing adversary cannot cancel out the honest parties contribution to $\hat{u}$, due to the ZKPs.

While this approach still involves ZKPs, it enjoys several benefits. First, the ZKPs are offloaded to an offline phase, making the online phase closer to being real-time. Second, the ZKPs prove statements over encryptions of zeros, independent of the inputs and circuits to be evaluated. Consequentially, they can be generated during idle periods of the system, and can be utilized during peak time. Third, these ZKPs can be batched (see e.g. [ABGS23]), which can significantly decrease the amortized overhead. Fourth, since the value $u + \hat{u}$ can be effectively perceived as a succinct proof of correct decryption, only one party has to combine the decryption shares during the online phase, which takes $O(N)$ time. As a result, when the network is receiving a batch of $\geq N$ of threshold decryption requests, the amortized cost of decryption per party is $O(1)$, independent of N.

Finally, as for (iii), we observe that the encryptions of the randomizers can be instantiated with any *threshold additively homomorphic* encryption (TAHE) scheme. By utilizing TAHE schemes with ciphertext size independent of N, (e.g. [BDO23] based on Class-Groups and [FMM+23] based on Paillier), we achieve $O(1)$ communication complexity per party.

Nevertheless, the drawback of the above approach is that it does not support identifying corrupted parties. We therefore require parties to provide ZKPs for partial decryption as before. If any of the verification steps in the key-switching-based solution fails, the protocol examines these ZKPs to identify the cheaters.

2 Preliminaries

2.1 Notation

The computational and statistical security parameters are denoted by κ and σ, respectively. We use $\mathrm{poly}(x)$ and $\mathrm{negl}(x)$ to represent generic polynomial and negligible functions in x, respectively. For an integers $m_1, m_2 \in \mathbb{Z}$, we denote by $[m_1, m_2] := \{m_1, \ldots, m_2\}$, and $[m_1] := [1, m_1]$ for brevity. For $a \in \mathbb{Z}$, we denote by $[a]_q$ the unique integer in $\mathbb{Z}_q$ with $[a]_q = a \mod q$. For a polynomial $g(x) = \sum_{i=0}^{\deg(g)} g_i x^i \in \mathbb{Z}[x]$ we denote $\|g\|_1 = \sum_{0 \leq i \leq \deg g} |g_i|$ and $\|g\|_\infty = \max_{0 \leq i \leq \deg g} |g_i|$. Given a polynomial $\Phi(x)$, we define a new norm $\|g\|_{\infty,\Phi} = \|g \mod \Phi\|_\infty$. Public encryption of a plaintext pt with public key pk is denoted by $\mathsf{ct} = \mathsf{E}(\mathsf{pk}, \mathsf{pt})$ and decryption of a ciphertext ct using secret key sk by $\mathsf{D}(\mathsf{sk}, \mathsf{ct})$. In a slight abuse of notation, we denote by $\mathsf{E}_{\mathsf{sk}}(\mathsf{pt}, a, e)$ the affine function $a \cdot \mathsf{sk} + \mathsf{pt} + e$.

2.2 Communication and Security Model

Communication. Our protocol is implemented over an *asynchronous reliable broadcast (ARB)* channel. Our DKG protocol is realized over a consensus protocol that is implemented on top of the ARB channel. In particular, the honest parties receive messages in the same *order*. Therefore, they can reach an *agreement* on the set of valid $t + 1$ messages sent in a previous round and the corresponding set of participants. This is written implicitly in the description of our DKG protocols.

Adversarial Model. The adversary statically corrupts up to $t < N$ parties, where $t + 1$ parties are required for threshold decryption. In addition, the adversary can *proactively* and *adaptively* block or delay messages. Due to theoretical limitations, the adversary may delay or block up to $f \leq N/3$ in each communication round. The adversary also has access to the headers of any communication between honest parties and ideal functionalities, and for ease of exposition, we assume this implicitly throughout without explicitly stating it each time.

We note that in case $f > N/3$, the adversary may fork the network into two disjoint components. This may result with a key-recovery attack. For instance, if the same preprocess round is used for two threshold decryption online rounds, the adversary will get linear equations over the secret key. Also, if $t+1 < N-f$, the adversary can DOS the system by blocking f parties. Therefore, typically one sets $t = \lfloor 2N/3 \rfloor$ and $f = \lfloor N/3 \rfloor$.

2.3 Homomorphic Commitments

Commitments are a cryptographic primitive that allows a party to commit to a specific value while keeping that value hidden. We refer to Appendix A for a formal definition based on [Gro09]. A commitment scheme is considered *homomorphic* if the message space $\mathcal{MS}_{\mathsf{pp}}$, randomness space $\mathcal{RS}_{\mathsf{pp}}$, and commitment space $\mathcal{CS}_{\mathsf{pp}}$ are equipped with an algebraic structure, often abelian groups. In this case, $\mathbf{com}$ serves as a homomorphism between $\mathcal{MS}_{\mathsf{pp}} \times \mathcal{RS}_{\mathsf{pp}}$ and $\mathcal{CS}_{\mathsf{pp}}$. Here, pp denotes the public parameter.

2.4 Zero Knowledge Proofs (ZKPs)

A *Zero Knowledge Proof (ZKP)* is a cryptographic technique that enables one party (the prover) to convince another party (the verifier) that a given statement is true, without disclosing any additional information beyond the truth of the statement itself. For formal definitions of ZKPs, we refer to [GO94].

In our `ThFHE` protocol, all statements can be framed as linear equations over a polynomial ring $\mathcal{R}_Q$ and a set of range claims. Namely:

$$\mathcal{L}_{\mathsf{lin}}^{A,\boldsymbol{B}} = \{(\boldsymbol{Y}; \boldsymbol{x}) \in \mathcal{R}_Q^{(\cdot)} \mid A\boldsymbol{x} = \boldsymbol{Y} \wedge \|\boldsymbol{x}\|_\infty \leq \boldsymbol{B}\},$$

where A is a matrix over $\mathcal{R}_Q$. For instance, the relation of a public-private key-pair $b = as + e$ can be expressed with $\boldsymbol{x} = (s, e)$ and $A = (a, 1)$, and the norms of the secret key s and noise e can be bounded. The same applies to public or symmetric encryption, the decryption shares, the key-switching key and the relinearlization key.

There are efficient constructions for proving a *batch* of such statements, with logarithmic proof size [DPLS19]. While their scheme is based on Bulletproofs [BBB+18] which is not PQ-secure, the work in [ABGS23] can be used alternatively. It only supports statements over $\mathbb{Z}_Q$ wherein $\boldsymbol{x}$ has ternary $\{-1, 0, 1\}$ coefficients, but this can be easily generalized by introducing the number of

modulations $\bmod \Phi$ as a witness (as internally done in [DPLS19]), and decomposing each witness in a ternary basis (similarly to the bit-decomposition used in [DPLS19]).

2.5 Polynomial Rings, Ring LWE, and Public-Key Encryption

2.5.1 Polynomial Rings. For a monic, irreducible polynomial $\Phi \in \mathbb{Z}[x]$ one can define a number field $K = \mathbb{Q}[x]/\Phi$ along with the corresponding polynomial ring $\mathcal{R} = \mathbb{Z}[x]/\Phi$. In cryptographic applications, it is common to consider the setting of cyclotomic fields where $K = \mathbb{Q}[x]/\langle \Phi_n(x) \rangle$ where $\Phi_n(x)$ is the n-th cyclotomic polynomial. Specifically, the case where $n = 2^d$, i.e., powers of two, is frequently used, and in this case $\Phi_n(x) = x^n + 1$ is a typical choice.

For an integer $Q \in \mathbb{Z}$, we can consider the modular ring $\mathcal{R}_Q = \mathcal{R}/Q\mathcal{R}$. The elements of $\mathcal{R}_Q$ can be represented as polynomials $a(x) = \sum_{j=0}^{n-1} a_j x^j$ of degree $n-1$, with coefficients $a_j \in \mathbb{Z}_Q$. This is referred to as the coefficient representation of the polynomial. When we refer to a ring element a, we usually mean its polynomial representation as a collection of coefficients $a = \{a_j\}_{j=0}^{n-1}$. For a ring $\mathcal{R}$, we denote by $\mathcal{R}^*$ the group of units (invertible elements) in $\mathcal{R}$.

2.5.2 Ring LWE (RLWE). At a high level, the Ring Learning with Errors (RLWE) assumption states that, for a suitably chosen Q and distribution χ, the following problem becomes computationally infeasible to solve in polynomial time relative to n. Consider the following experiment:

- Generate a set of uniform ring elements $a^{(i)} \in \mathcal{R}_Q$, where $i \in [m]$. These elements are obtained by uniformly sampling the coefficients of a polynomial.
- Select a ring element $s \in \mathcal{R}_Q$ (referred to as the "secret").
- Sample ring elements $e^{(i)} \in \mathcal{R}$ from the distribution χ for $i \in [m]$.

Define $b^{(i)} = a^{(i)}s + e^{(i)}$, and consider the sequence of *samples* $(a^{(i)}, b^{(i)})_i$. For $m = \text{poly}(n)$, distinguishing this sequence from one where the $b^{(i)}$ are uniformly sampled is computationally indistinguishable, provided that the distinguishing process has polynomial complexity $\text{poly}(n)$.

2.5.3 Public-Key Encryption. We use the BGV HE scheme [BGV14] and provide an overview of its construction in Figure 6 (Appendix A). Note that while we do not specify $\cdot(x)$ all sampled values including p and m can be chosen as polynomials (in $\mathbb{Z}[x]$ and $\mathbb{Z}[x]/p$ respectively). This work details the adaptation of our `ThFHE` protocol to implement a threshold version of the BGV scheme. Additionally, our approach can be extended to support other schemes such as FV [FV12] and CKKS [CKKS17], which have similar structures for secret, public, and re-linearization keys. For instance, the re-linearization key in FV is defined as $\mathsf{lk} = (a_0, [-a_0 \cdot s + e_0 + s^2]_Q)$. Furthermore, our method can also be adapted to include Torus-FHE, with details provided in Appendix F.

2.6 Shamir Secret Sharing over Polynomial Rings

A Secret Sharing Scheme is a cryptographic method for distributing a secret among multiple parties such that the secret can only be reconstructed when a sufficient number of shares are combined. Our scheme employs Shamir Secret Sharing over polynomial rings, which is applicable to any commutative ring. We denote by $\lambda^S_{\alpha,\beta} := \prod_{\alpha' \in S \setminus \{\alpha\}} \frac{\beta - \alpha'}{\alpha - \alpha'}$ the Lagrange interpolation coefficient at point β with set $S \subset \mathcal{R}$ using $\alpha \in S$. That is to say, for every polynomial f of degree t over $\mathcal{R}$ and a proper set S (see below) of $t+1$ interpolation points, the following equation holds $f(\beta) = \sum_{\alpha \in S} \lambda^S_{\alpha,\beta} f(\alpha)$. We will use this over polynomial rings, and so the polynomial ring elements $\alpha(x), \beta(x) \in \mathcal{R} = \mathbb{Z}[x]/\Phi(x)$ may themselves be represented as a polynomials.

Since the Lagrange coefficient formula involves fractions, in order for them to be well-defined and for reconstruction to be possible, the set of interpolation points must be an *exceptional set* [ACD+19]:

Definition 1 ([ACD+19] (Exceptional Set)). *A set $\mathcal{I} = \{\alpha_1, \ldots, \alpha_N\} \subseteq \mathcal{R}$ of ring elements is an* exceptional set *if for each $\alpha_i \neq \alpha_j \in \mathcal{R}^*$ have that $\alpha_i - \alpha_j \in \mathcal{R}^*$. The* size *of an exceptional set is defined as $B_{\mathcal{I}} := \max(\left\|\alpha_i^k\right\|_{\infty,\Phi} : k \in \{\pm 1\}, i \in [N])$.*

A (t, N) Shamir secret sharing over a polynomial ring $\mathcal{R}$ may be defined as follows. Let $\mathcal{I} = \{\alpha_1, \ldots, \alpha_N\}$ be an exceptional set. Given a secret $s \in \mathcal{R}$, the *dealer* samples $r_i \leftarrow \mathcal{R}$ and computes the polynomial $f(X) = s + \sum_{i=1}^{t} r_i X^i : \mathcal{R} \to \mathcal{R}$. It then sends to each party P_i, associated with a unique non-zero element α_i from the exceptional set $\mathcal{I}$, its share, defined as $[s]_i = f(\alpha_i)$. To reconstruct the secret s, a subset $S \in \binom{[N]}{t+1}$ can use Lagrange interpolation on the shares they possess, namely: $\sum_{i \in S} \lambda^S_{\alpha_i,0} \cdot s_i = s$.

2.7 Verifiable Secret Sharing (VSS)

A secret sharing scheme is termed Verifiable Secret Sharing (VSS) if it satisfies two properties: (i) *fairness*, which ensures that a malicious dealer cannot generate verified shares that lead to different secrets for different subsets of parties. In other words, all subsets reconstruct the same secret; (ii) *secrecy* , which guarantees that no information about the secret is revealed during the protocol. We adapt these definitions from [Fel87, Sha79]. For a more comprehensive understanding of Shamir secret sharing over a ring, we refer the reader to [Feh98].

Definition 2 ([Fel87]). *A Verifiable Shamir secret sharing scheme is considered* fair *if, for every polynomial-time dealer $\mathcal{D}$, it is impossible to produce shares s_i such that $\sum_{i \in S} \lambda^S_{0,i} s_i \neq \sum_{i \in S'} \lambda^{S'}_{0,i} s_i$ for two different sets $S, S' \in \binom{[N]}{t+1}$ consisting only of verified shares.*

Definition 3 ([Sha79]). *We say that a secret sharing scheme is t-secret if for any set of corrupted parties U such that $|U| \leq t$, any two secrets $s, s' \in \mathcal{R}$ and independent random coins, the distributions of the resulting private shares $\{s_i\}_{i \in U}$ and $\{s'_i\}_{i \in U}$, are statistically indistinguishable.*

2.8 Publicly Verifiable Secret Sharing (PVSS)

A VSS scheme is *public* if any party can check the validity of the sharing and reconstruction phase. Typically, the dealer publishes encrypted shares of a secret so that parties holding the corresponding decryption keys may later reconstruct it. The encryptions are attached with proper ZKPs that enable verification of their validity by any party, including parties that did not receive any share. Both dealing and reconstruction are non-interactive, which is useful for *distributed key generation (DKG)* protocols, especially under asynchronous broadcast-only communication channels [CD24]. In the context of threshold decryption it is enough to have public verifiability on the sharing as the secret itself is never fully reconstructed. Thus we only present the parts of the formalism which will be relevant to this paper.

Definition 4. *(PVSS [CD24]) A* PVSS *scheme includes the following:*

- *Setup:* $\mathsf{pp} \leftarrow \mathit{Setup}(1^\kappa, \mathit{aux})$ *outputs public parameters* pp *given a security parameter and auxiliary data, specifying the number of parties* N*, reconstruction threshold* t*, and the space of secrets* $\subseteq \mathcal{R}$*. In addition, each party* $P_i \in \mathcal{P}$ *is associated with a public encryption key* pk_i*, attached with a ZKP* Π^i_{pk} *of knowledge of a corresponding secret decryption key* sk_i*.*
- *Secret distribution:* $(\{\mathsf{ct}_i\}_{i\in[N]}, \{F_k\}_{k\in[t]}, \Pi_{\mathit{share}}) \leftarrow \mathit{Dist}(\mathsf{pp}, s)$*, computes a secret sharing* $[s]$ *of* s*, and outputs encrypted shares* $\mathsf{ct}_i = \mathsf{E}_{\mathsf{pk}_i}([s]_i)$*, commitments on the secret polynomial* F*, along with a proof that they are correct* Π_{share}*. The verification key of party* i *is then homomorphically evaluated as* $\mathsf{vk}_i = \sum_k F_k \alpha_i^k = \mathbf{com}(f(\alpha_i))$*.*
- *Decryption Share:* $[s]_i \leftarrow \mathit{DecShare}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct}_i)$*, outputs a secret share* $[s]_i = \mathsf{D}_{\mathsf{sk}_i}(\mathsf{ct}_i)$*. It can be self-verified as it opens the commitment* $\mathsf{vk}_i = \mathbf{com}([s]_i)$*.*
- *Reconstruction:* $(s') \leftarrow \mathit{Rec}(\mathsf{pp}, \{[s]_i\}_{i\in S})$ *reconstructs the secret given a set* S *of* $t+1$ *decryption shares.*

We refer the reader to [CD17,CD24] for further details about the game based security definitions of a PVSS scheme. We will used the IND1 variant from [CD17] as presented in Appendix A.

3 System Overview

In this section, we cover in a high-level the components of a distributed system that is based on a `ThFHE` scheme. The system, depicted in Figure 1, is composed of four entities: (i) *clients*, (ii) *evaluators*, (iii) *decryptors*, and (iv) a *distributed ledger*. Essentially, the clients, each holding some private data pt_i, want to aggregate their inputs and learn a common output $y = f(\mathsf{pt}_1, \ldots)$. The evaluators are responsible for the computation of f, which might also be private.[5] The decryptors are responsible for decoding and retrieving the output y.

[5] The evaluators may be realized by a service provider, in which case f may be a private IP.

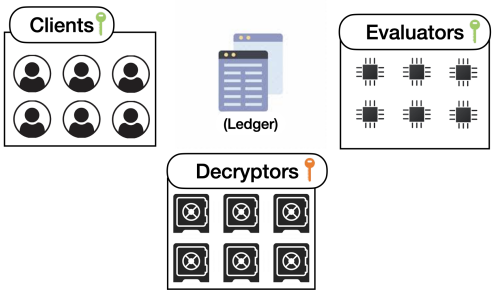


Fig. 1: A potential design of a distributed system based on `ThFHE` scheme, as described in Section 3.

First, the set of decryptors D participate in a *Distributed Key Generation (DKG)* protocol. This is a distributed setup protocol that is responsible for generation of a public encryption key pk, a verifiable threshold secret sharing of the corresponding private key $[\mathsf{sk}], [\mathsf{pk}]$, as well as an evaluation key evk used for homomorphic evaluation. After the DKG is done, the clients $\{C_i\}_i$ encrypt their private inputs $\mathsf{ct}^i_{\mathsf{in}} = \mathsf{E}(\mathsf{pk}, \mathsf{pt}_i)$ using the public key pk, and broadcast it, e.g. by posting it on a distributed ledger. This of course, keeps their data private. Then, each currently available evaluator E_j uses the evaluation key evk to homomorphically aggregate and evaluate the function f, deriving $\mathsf{ct}^j_{\mathsf{out}} = \mathsf{EV}(\mathsf{evk}, \mathbf{ct}_{\mathsf{in}}, f)$, and broadcasts it on the ledger.[6] Then, the decryptors invoke a one-round *threshold decryption* protocol that works as follows. Each currently available decryptor D_k applies a majority vote to determine the output ciphertext to decrypt $\mathsf{ct}_{\mathsf{out}}$. Each decryptor uses its secret key share sk_k to compute and post a decryption share $\mathsf{ds}_k = \mathsf{PD}(\mathsf{ct}_{\mathsf{out}}; \mathsf{sk}_k)$, and potentially adds a ZKP of correctness of the decryption share, which binds the decryption share ds_k to $(\mathsf{vk}_k, \mathsf{ct}_{\mathsf{out}})$. The decryption shares are then aggregated to derive the plaintext results y. Since this is public computation, it can be conducted by the evaluation parties and posted on the ledger, from which clients may retrieve the majority output.

Keeping the above system in mind, we recognize that large-scale applications should tolerate congestion spikes, potentially by exploiting idle periods for pre-processing. To this end, we also propose a pre-processing routine, independent of client encrypted inputs, that can be executed by the decryptors in idle periods. Upon receiving decryption requests, that pre-processed can be used to enhance the performance of the threshold decryption phase. In particular, the decryption output can be verified without verifying each decryption share ZKP. As a result, instead of letting each evaluator to compute each plaintext, this work can be evenly split among them, yielding an amortized cost of $\mathcal{O}(|D|/|E|)$ for combining the decryption shares.

[6] Alternatively, a service provider may verifiably compute $\mathsf{ct}_{\mathsf{out}}$, by attaching a SNARK [BCCT12] proof of correct computation.

3.1 Ideal ThFHE Functionality

In this section, we present the *ideal threshold FHE functionality* $\mathcal{F}_{\texttt{ThFHE}}$, which is outlined in Functionality 1. The functionality is composed of two sub-routines: key generation, which is realized by a DKG protocol, and decryption, realized by an asynchronous threshold decryption protocol. In addition we give an additional functionality $\mathcal{F}_{\texttt{ThFHE}}^{\mathsf{preprocess}}$ (Functionality 2) which is analogues to $\mathcal{F}_{\texttt{ThFHE}}$ but uses a decryption which reveals the encryption randomizer.

FUNCTIONALITY 1. (*ThFHE- $\mathcal{F}_{ThFHE}$*)

PARAMETERS: a set of N parties $\mathcal{P} = \{P_i\}_{i\in[N]}$, a threshold $t \leq N/3$, an adversary $\mathcal{A}$ controlling a subset of the parties $\mathcal{P}_{\mathcal{A}} \subset \mathcal{P}$ ($|\mathcal{P}_{\mathcal{A}}| \leq t$), an encryption scheme $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ with an *affine key-homomorphism*.
BEHAVIOR:

1. **Setup**: Upon receiving a command $(\textbf{keygen}, \mathsf{sid}, P_i)$ from party $P_i \in \mathcal{P}$, send to $\mathcal{A}$ and record $(\textbf{keygen}, \mathsf{sid}, P_i)$. After recording $t+1$ requests for a given sid, generate a key triplet $(\mathsf{pk}, \mathsf{evk}; \mathsf{sk}) \leftarrow \mathsf{G}(1^\kappa)$, and send $\mathsf{pk}, \mathsf{evk}$ to $\mathcal{A}$.
   - Upon receiving $(\textbf{bias}, \mathsf{sid}, \delta, \varepsilon)$ from $\mathcal{A}$, set $\mathsf{sk} \leftarrow \delta \cdot \mathsf{sk} + \varepsilon$, update the public key pk and evk accordingly.
   - Record $(\mathsf{sid}, \mathsf{pk}, \mathsf{evk}; \mathsf{sk})$ and send $(\mathsf{pk}, \mathsf{evk}, \|\delta\|_\infty, \|\varepsilon\|_\infty)$ to all parties.
2. **Decryption**: Upon receiving a command $(\textbf{decrypt}, \mathsf{ssid}, \mathsf{ct}, P_i)$, for $\mathsf{ssid} = (\mathsf{sid}, \ldots)$, and P_i that is not recorded, if there is a record of $(\mathsf{sid}; \ldots, \mathsf{sk})$, send to $\mathcal{A}$ and record $(\mathsf{ssid}, \mathsf{ct}, P_i)$. Upon recording $t+1$ requests, compute the plaintext $\mathsf{pt} = \mathsf{D}(\mathsf{sk}, \mathsf{ct})$. Then, broadcast pt to all parties.

CCA Security. As FHE schemes cannot be CCA secure, if an adversary manages to decrypt an adversarially chosen ciphertext, it may extract the secret key sk. However, the functionality decrypts only after $t+1$ parties request decryption of a ciphertext, which must include one honest party. Therefore, as long as honest parties validate that the ciphertexts are safe to decrypt, this will not lead to a chosen ciphertext attack. In the system described above, input ciphertext should be attached with zk-proofs of well-formedness, and a threshold of t_e+1 evaluators should be required to accept a ciphertext for decryption. Note that indeed, if the adversary controls more than t_e evaluators, or more than t decryptors, it will be able to extract sk.

RKA Security. Our functionality allows the adversary to add a multiplicative and additive bias to the secret key. This is a requirement of our asynchronous DKG protocol, see discussion in Section E.5. In turn, we require the encryption scheme to be *key-homomorphic*, which ensures semantic security even under Related Key Attacks (RKAs) [AHI10].

In the following sections, we focus on realizing the decryption part of the functionality.

4 Shamir Secret Sharing Over Rings

In this section, we present an adaptation of Shamir's Secret Sharing scheme [Sha79] over finite prime fields. Our adaptation is somewhat captured in [DSDFY94], who generalized Shamir Secret Sharing to carry over polynomial rings and modules. We adapt their approach, in order to make it compatible with asynchronous lattice-based `ThFHE` schemes. Intuitively, under this setting, as observed in [BGG+18], it is also important to keep the interpolation points and coefficients small. Essentially, the secret sharing scheme over $\mathcal{R}_Q$ should minimize the norm of the following parameters:

- The size $B_{\mathcal{I}}$ of the exceptional set of interpolation points $\mathcal{I}$ (see Lemma 2). This affects the security proof of threshold decryption.
- The denominator clearing factor Δ, the Lagrange coefficients at zero λ_0, and the products $\lambda\Delta$ (see Corollary 1 and Lemma 4). This is essential for both the correctness and security of the threshold decryption algorithm.

Remark 1. Our proposed instantiation exploits the specific structure a the power-of-two *cyclotomic* ring $\mathcal{R}_Q = \mathbb{Z}_Q[x]/\Phi_{2n}(x)$ where $\Phi_{2n}(x) = x^n + 1$. Nonetheless, the vast majority of RLWE-based schemes work over cyclotomic rings.[7]

4.1 Instantiation of $\mathcal{I}$

Next, we propose a concrete instantiation for the set of interpolation points $\mathcal{I}$ and later for the denominator clearing factor Δ. We assume $N < n$, which we argue to be reasonable for most settings (e.g., typically security requires $n > 1000$, and $N \leq 1000$). Also for ease of exposition, we take $N \equiv 0 \mod 6$. The set of interpolation points is then fixed as $\mathcal{I} = \{(-1)^i x^j : (i,j) \in \{0,1\} \times [\frac{N}{2} - 1]\}$. Looking ahead, each party $P_{i,j}$ will be associated with the interpolation point $(-1)^i x^j$. Given Lemma 1, we show that $\mathcal{I}$ is an exceptional set of minimal size in Lemma 2.

Lemma 1 ([ACD+19]). *Let $\Phi_{2n}(x) = x^n + 1$ be the power-of-two cyclotomic polynomial. Let $g \in \mathbb{Z}[x]$ be a polynomial of bounded degree $\deg(g) \leq M(n-1)$ for $M \geq 1$. Then $\|g\|_{\infty,\Phi} \leq M\|g\|_\infty$.*

This lemma allows us to essentially operate within $\mathbb{Z}[X]$ and subsequently estimate the norms in our ring.

Lemma 2. *If Q is odd, then $\mathcal{I}$ is an exceptional set of size $B_{\mathcal{I}} = 1$.*

Proof. According to Definition 1, our goal is to prove that the difference between any two points $(i,j) \neq (i',j')$ in $\mathcal{I}$ is invertible. To achieve this, we examine the following cases.

- If $j = j'$, we obtain an element of the form $\pm 2x^j$. Since Q is odd, 2 is invertible. Also x^j is invertible in $\mathcal{R}$, as $x^j \cdot x^{n-j} = x^n = -1 \mod \Phi_{2n}(x)$.

[7] Our instantiation can be generalized to work with normed, $\mathbb{Z}$-graded rings [Mar93].

– If (WLOG) $j > j'$, and $i = i'$, we have $\pm(x^j - x^{j'}) = \pm x^{j'}(x^{j-j'} - 1)$ which requires us to verify that $x^{j-j'} - 1$ is invertible.
– Similarly, if $j > j'$ and $i \neq i'$, we need to show that $x^{j-j'} + 1$ is invertible.

It remains to show that the expressions $x^j \pm 1$ for all $j \in [N/2]$ are invertible. Indeed, as $(x-1)(1 + x + \cdots + x^{n-1}) = x^n - 1 \equiv -2$, we get that the inverse of $(x-1)$ is exactly $-2^{-1}(\sum_{i=1}^{n} x^i)$. By replacing x in this equation with $-x$ we obtain that $x + 1$ is invertible as well. Lastly, by substituting $x \leftarrow x^j$, we can conclude that $x^j \pm 1$ is always invertible. As for the size of $\mathcal{I}$, we have $\left\|\pm x^i\right\|_{\infty,\Phi} = 1$. Furthermore, $\left\|(\pm x^i)^{-1}\right\|_{\infty,\Phi} = \left\|\mp x^{n-i}\right\|_{\infty,\Phi} = 1$, as required. □

4.2 Instantiation of Δ

Next, we propose a denominator clearing factor Δ with respect to $\mathcal{I}$ defined above. Recall that its purpose is to cancel out the denominator terms of all Lagrange interpolation coefficients. Therefore, we first show that indeed $\Delta \cdot \lambda^S_{(i,j),0}$ are all integral over $\mathbb{Z}[x]$, and bound their norms. This is done in Theorem 1, a crucial technical theorem in this paper. Then, we use techniques from [ACD+19] in order to bound their norms in the cyclotomic ring $\mathcal{R}$.

First, let us briefly discuss the rationale the lead us to the selection of Δ. Seemingly, the first choice that comes to mind is $\Delta_0 = (\prod_{e=1}^{\frac{N}{2}} x^{2e} - 1)^2$. The idea being that, the term $x^e - 1$ may appear only twice per each Lagrange coefficient, as we must have $|j - j'| = e$ and $i = i'$. Similarly, the term $x^e + 1$ only appears for $|j - j'| = e$ and $i \neq i'$.

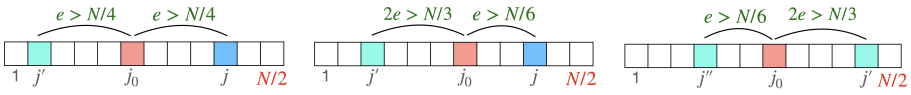
This can be further refined by recognizing that the term $\pm x^e - 1$ may appear twice only when $e \leq \frac{N}{4}$, as otherwise $|j - j'| = e$ has only one solution. This leads to $\Delta_1 = (\prod_{e=1}^{\frac{N}{2}} x^{2e} - 1)(\prod_{e=1}^{\frac{N}{4}} x^{2e} - 1)$, since only the lower half of the possible exponents should appear twice in the multiplication.

Finally, we observe that even after removing the last $N/12$ terms from Δ_1, there are still enough terms left over in order to cancel-out all denominators. Essentially, we exploit that fact that the terms of the form $x^e \pm 1$ can be canceled out by both $x^{2e} - 1$ and $x^{4e} - 1$. Formally,

Theorem 1. *Let* $\Delta = 2 \prod_{e \in [\frac{N}{2}]} (x^{2e} - 1) \prod_{j \in [\frac{N}{6}]} (x^{2e} - 1)$. *Then for every subset* $S \subset \{0,1\} \times [\frac{N}{2} - 1]$ *such that* $1 < |S| < N$ *and every distinguished element* $(i_0, j_0) \in S$, *(1)* $\Delta\lambda^S_{(i_0,j_0),0} \in \mathbb{Z}[x] \subset \mathbb{Q}[x]$ *is a polynomial; and (2)* $\left\|\Delta\lambda^S_{(i_0,j_0),0}\right\|_1 \leq 2^{\frac{3}{4}N}$.

Proof. Recall that the Lagrange interpolation coefficient at the point 0 with the set S can be expressed as follows:

$$\lambda^S_{(i_0,j_0),0} = \frac{(-1)^{\sum_{(i,j)\in S\setminus\{(i_0,j_0)\}} i} x^{\sum_{(i,j)\in S\setminus\{(i_0,j_0)\}} j}}{\prod_{(i,j)\in S\setminus\{(i_0,j_0)\}} (-1)^i x^j - (-1)^{i_0} x^{j_0}}$$



(a) Stage 2a (b) Stage 2c (c) Stage 2c

Fig. 2: Double cancellation analysis.

First, if $(1 - i_0, j_0) \in S$, the corresponding term in the denominator would be $\pm 2x^{j_0}$, which is canceled-out with the factor 2 in Δ. We therefore assume WLOG that it is not the case. In addition, we disregard the numerator, as it has no bearing on the integrality of the expression, and is of norm one. Indeed, if we denote,

$$\Pi_1 := \prod_{e \in [\frac{N}{2}-1]} (x^{2e} - 1), \qquad \Pi_> := \prod_{(i,j) \in S \setminus \{(i_0,j_0)\} : j > j_0} x^{j-j_0} + (-1)^{i_0+i+1}$$

$$\Pi_2 := \prod_{e \in [\frac{N}{6}]} (x^{2e} - 1), \qquad \Pi_< := \prod_{(i,j) \in S \setminus \{(i_0,j_0)\} : j < j_0} x^{j_0-j} + (-1)^{i_0+i+1}$$

Then, we have:

$$\pm \Delta \lambda^S_{(i_0,j_0),0} = x^c \frac{\Delta}{\Pi_> \Pi_<} = x^c \frac{2\Pi_1 \Pi_2}{\Pi_> \Pi_<}$$

where $c := \sum_{(i,j) \in S \setminus \{(i_0,j_0)\} : j > j_0} j$. To prove the first statement of the theorem (1), we need to show that we can cancel all terms in both $\Pi_<$ and $\Pi_>$, using Π_1 and Π_2. First, we will provide a summary of our approach, followed by an explanation of its effectiveness. Our method operates in two stages:

1. If a term $x^e \pm 1$ appears in the product $\Pi_<$, we cancel it with the corresponding term $x^{2e} - 1 = (x^e - 1)(x^e + 1)$ in Π_1.
2. If a term $x^e \pm 1$ appears in the product $\Pi_>$, we proceed differently depending on the value of e and the outcome from stage (1).
   (a) If $e \geq \frac{N}{4}$, we cancel it using the term $x^{2e} - 1$ from Π_1.
   (b) If $e \leq \frac{N}{6}$, we cancel it using the term $x^{2e} - 1$ from Π_2.
   (c) If $\frac{N}{6} < e < \frac{N}{4}$, and the term $x^{2e} - 1$ in Π_1 has not been canceled yet, we cancel it there; otherwise, we cancel the term $x^{4e} - 1$ in Π_1.

Next, we prove that no term in the numerator is used twice.

1. At the beginning of stage 1, no terms have been canceled yet. Moreover, note that in both $\Pi_>$ and $\Pi_<$, each term $x^e \pm 1$ may appear only once, when the terms are not factored. For instance, we cannot have $(x^2 - 1)^2$, but we may have $(x^2 - 1)(x^4 - 1) = (x^2 - 1)^2(x^2 + 1)$. Lastly, since $1 \leq e \leq \frac{N}{2} - 1$, the term $x^{2e} - 1$ is indeed present in Π_1.
2. Next, we show that there are no double cancellations at stage 2.

(a) For demonstration, consider Figure 2(a). Since $e \geq \frac{N}{4}$, the equation $|j - j_0| = e$ has at most one solution for $j \in [\frac{N}{2}]$. Because there is a solution with $j > j_0$ (in blue), there is no solution with $j < j_0$ (j' in turquoise). Thus, the term $x^{2e} - 1$ from Π_1 was not used in stage (1). Furthermore, as each term in $\Pi_>$ appears at most once, we do not cancel the same numerator term more than once in this stage.

(b) By this stage, we have not utilized any terms from Π_2. Additionally, since $\Pi_>$ has no duplicates, we cannot cancel the same term twice.

(c) We show that $x^{4e}-1$ could not have been canceled in any of the previous stages. First, assume it was canceled in stage (1), as in Figure 2(b). Then $x^{2e} \pm 1$ appeared in $\Pi_<$ (j' in turquoise), and therefore, $j_0 > 2e > N/3$. But since $x^e \pm 1$ appears in $\Pi_>$ (j in blue), we know that $N/2 - j_0 > e > N/6$. Combining the two, we get $N/2 = j_0 + (N/2 - j_0) > N/3 + N/6 = N/2$, a contradiction. Otherwise, assume the term $x^{4e} - 1$ from Π_1 was already used at stage (2). This is depicted in Figure 2(c). It must have been used at stage (2a), as in stage (2b) only terms from Π_2 are used. In particular, the term $x^{2e} \pm 1$ must have appeared in $\Pi_>$ (j' in turquoise), which means that $N/2 - j_0 > 2e > N/3$. In addition, we know that the term $x^{2e} - 1$ in Π_1 was already canceled. If it was canceled with an element from $\Pi_<$ (j'' in turquoise), then $j_0 > e > N/6$, and we get $N/2 = N/2 - j_0 + j_0 > N/3 + N/6 = N/2$, a contradiction.
Therefore, we deduce that it was canceled by an element from $\Pi_>$. But since there is only one element in $\Pi_>$ of the form $x^e + 1$ and of the form $x^e - 1$, the term $x^{2e} - 1$ can be used to cancel out both of them.

Next, we turn to the evaluation of the norm required to prove (2). First, observe that multiplying by x^c does not affect the ℓ_1-norm of a polynomial. We will establish a bound on the norm by counting the number of terms and utilizing the sub-multiplicativity of the ℓ_1-norm.

If a term was canceled in Stages 1, (2a), or (2b), the total number of numerator terms may only decrease. These cancellations take the form $(x^{2e} - 1) \to x^e \pm 1$ or $x^e \pm 1 \to 1$, where we use the notation $\to$ to indicate "turns into".

Finally, if a term was canceled in stage (2c), it could introduce a single term to the numerator as follows: $x^{4e} - 1 \to (x^{2e} + 1)(x^e \pm 1)$. This can happen for at most $\frac{N}{4} - \frac{N}{6} - 1$ terms. Therefore, the total number of terms we have in $\frac{\Pi_1\Pi_2}{\Pi_<\Pi_>}$ is at most $\frac{N}{2} + \frac{N}{6} + \frac{N}{4} - \frac{N}{6} - 1 = \frac{3}{4}N - 1$. Each term is of the form $x^e \pm 1$, which has a ℓ_1-norm of 2. This allows us to bound the norm by $\frac{2^{\frac{3}{4}N}}{2}$. Recalling that $\Delta = 2\Pi_1\Pi_2$, the overall bound is $2^{\frac{3}{4}N}$. □

In what follows, we state the necessary upper bounds to complete our construction. The proofs are are provided in Section E.1.

Lemma 3. $\|\Delta\|_\infty \leq 1.2^{\frac{2}{3}N}$

Corollary 1. *With the same notations as above, we have (1)* $\|\Delta\|_{\infty,\Phi_n} \leq \frac{N^2}{3n} 1.2^{\frac{2}{3}N}$*; and (2)* $\left\|\Delta \cdot \lambda^S_{(i_0,j_0),0}\right\|_{\infty,\Phi_n} \leq \frac{N^2}{3n} 2^{\frac{3}{4}N}$.

Lemma 4. *For any* $U \in \binom{[N]}{t}$ *we have that* $\left\|\lambda^{U\cup\{0\}}_{0,(i_0,j_0)}\right\|_{1,\Phi} \leq 2^t$.

5 Threshold Decryption

Our Protocol. We refer to Appendix C for a proposal of a DKG protocol that UC realizes the setup phase of Functionality 1. In what follows, we focus on threshold decryption. First, we note that the encryption process also has to be adjusted. The encryption process is similar to the classic HE scheme, but instead of adding the error term $e \leftarrow \chi_{\mathsf{E}}$, the encrypting party adds $\Delta \cdot e$. That is, the error term is multiplied by the denominator clearing factor from Section 4.

Furthermore, to align with our Shamir secret sharing scheme from Section 4, we re-index the set of decrypting parties by using elements of $\{0,1\} \times [\frac{N}{2} - 1]$. Namely, $\mathcal{P} = \{P_{0,0}, P_{1,0}, P_{0,1}, P_{1,1}, \ldots, P_{0,\frac{N}{2}-1}, P_{1,\frac{N}{2}-1}\}$. The error domains for encryption and distributed decryption, denoted χ_{E} and χ_{D}, are both uniform distributions over a sphere defined by the norm $\|\cdot\|_{\infty,\Phi_n}$ with radii $r_{\mathsf{E}}, r_{\mathsf{D}}$, respectively. Protocol 1 formally presents the construction of our `ThFHE` scheme. Notably, the probability of zero flooding is negligible due to the statistical security parameter σ in the condition on χ_D. We denote by Π^{enc} a zk proof of correct encryption and Π^{ds} a zk proof of correct decryption share generation.

PROTOCOL 1 (*ThFHE Construction* (Π_{ThFHE}))

1. **Key Generation**: The parties execute Π_{DKG} (Protocol 5). Each party $P_i \in \mathcal{P}$ receives the secret share sk_i of sk, the public key $\mathsf{pk} = (a, b)$ where $a \leftarrow \mathcal{R}_Q$ and $e \leftarrow \chi_{\mathsf{E}}$, and $b = \mathsf{E}_{\mathsf{sk}}(0, a, p\Delta e)$, and the set of verification keys $\{\mathsf{vk}_j = \mathsf{COM}_{\mathsf{async}}(\mathsf{sk}_j)\}_{P_j \in \mathcal{P}}$ of all parties. The protocol may also output a homomorphic evaluation key evk.
2. **Encryption**(pt):
   (a) Sample $\tilde{s}, \tilde{e}_0, \tilde{e}_1 \leftarrow \chi_{\mathsf{E}}$.
   (b) Compute $\mathsf{ct}_0 = \mathsf{E}_{\tilde{s}}(\mathsf{pt}, b, p\Delta\tilde{e}_0)$ and $\mathsf{ct}_1 = \mathsf{E}_{\tilde{s}}(0, a, p\Delta\tilde{e}_1)$ and set $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$.
   (c) Generate a non-interactive ZK proof $\Pi^{\mathsf{enc}}(\mathsf{ct}; \mathsf{pt}, \tilde{s}, \tilde{e}_0, \tilde{e}_1)$ of correct encryption and broadcast $(\mathsf{ct}, \Pi^{\mathsf{enc}})$.
3. **Distributed Decryption**(ct):
   (a) Upon receiving the ciphertext $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$, each party $P_i \in \mathcal{P}$:
      – Samples $e_i \leftarrow \chi_{\mathsf{D}}$ and computes the decryption share $\mathsf{ds}_i = \mathsf{ct}_1 \cdot \mathsf{sk}_i + p\Delta e_i$.
      – Computes a ZK proof $\Pi^{\mathsf{ds}}_i(\mathsf{ct}, \mathsf{ds}_i, \mathsf{vk}_i; \mathsf{sk}_i, e_i)$ proving the correctness of ds_i and broadcasts $(\mathsf{ds}_i, \Pi^{\mathsf{ds}}_i)$.
   (b) Upon receiving $(\mathsf{ds}_j, \Pi^{\mathsf{ds}}_j)$ from another party P_j, each party P_i verifies the proof $\Pi^{\mathsf{ds}}_j(\mathsf{ct}, \mathsf{ds}_j, \mathsf{vk}_j)$. If invalid, ignore the message, and otherwise record it.
   (c) Upon recording $t + 1$ decryption shares from a subset S_i of parties, computes and broadcasts $\mathsf{pt} = \mathsf{ct}_0 - \sum_{j \in S_i} \lambda^{S_i}_{j,0} \mathsf{ds}_j \mod p$.

The correctness and security of the above construction are stated in Theorems 2,3 below, and their proofs are in Appendix E.7.

Theorem 2. *Assume that the distributions $\chi_{\mathsf{E}}, \chi_{\mathsf{D}}$ are supported on balls of radii $r_{\mathsf{E}}, r_{\mathsf{D}}$ respectively, with respect to the $\|\cdot\|_{\infty,\Phi}$ norm. If ct is an encryption of pt with error of size at most B and plaintext modulus p, then threshold decryption with Π_{ThFHE} (Protocol 1) will output pt if $\frac{Q}{p} > nN\lceil\frac{N^2}{n}\rceil(r_{\mathsf{D}}2^{\frac{3}{4}N} + B \cdot 1.2^{\frac{2}{3}N})$.*

Theorem 3. *Assume that the distributions $\chi_{\mathsf{E}}, \chi_{\mathsf{D}}$ are supported on balls of radii $r_{\mathsf{E}}, r_{\mathsf{D}}$ respectively, with respect to the $\|\cdot\|_{\infty,\Phi}$ norm. If $r_{\mathsf{D}} > n2^t2^\sigma r_{\mathsf{E}}$ then Π_{ThFHE} (Protocol 1) UC-realizes the ideal functionality $\mathcal{F}_{\mathit{ThFHE}}$.*

Remark 2. Letting $\chi'_{\mathsf{E}} = \alpha\chi_{\mathsf{E}}$ for a ring element $\alpha \in \mathcal{R}$ which is coprime to q, it holds that RLWE with noise sampled from χ'_{E} is equivalent to RLWE with noise sampled from χ'_{E}. A formal proof is given in Section E.2. In our case, Δ and Φ share no common roots over $\mathbb{C}$, as the roots of Δ are roots of unity of order strictly smaller than the roots of unity of Φ. Therefore, $\langle \Delta p, q\rangle = \langle 2p, q\rangle = 1$.

6 Leveraging Preprocessing Techniques

Our Construction. Consider a ciphertext $\mathsf{ct}' = (\mathsf{ct}'_0, \mathsf{ct}'_1)$ that satisfies:

$$\mathsf{ct}'_0 + \mathsf{ct}'_1 \cdot \mathsf{sk} \equiv \mathsf{pt} + pe' \bmod Q$$

with some small noise e'. We now convert ct' into a "committed" ciphertext ct such that (i) ct is still the encryption of pt. For ease of exposition, we consider that ct can be decrypted using the same sk; (ii) the encryption of the randomizer u associated with ct can be homomorphically evaluated. A threshold of parties could decrypt this u, which "decommits" the ciphertext ct and allows extraction and validation of the underlying plaintext.

For (i), we consider the key switching key ks from sk to itself as an encryption of sk, namely $\mathsf{ks} = (u_{\mathsf{ks}}b + pe_0^{\mathsf{ks}} + \mathsf{sk}, u_{\mathsf{ks}}a + pe_1^{\mathsf{ks}})$ for $u_{\mathsf{ks}}, e_0^{\mathsf{ks}}, e_1^{\mathsf{ks}} \leftarrow \chi_{\mathsf{E}}$, and set:

$$\mathsf{ct} = (\mathsf{ks}_0\mathsf{ct}'_1 + \mathsf{ct}'_0, \mathsf{ks}_1\mathsf{ct}'_1)$$

We have:

$$\mathsf{ct}_1 \equiv (u_{\mathsf{ks}}\mathsf{ct}'_1)a + pe_1^{\mathsf{ks}}\mathsf{ct}'_1$$

and

$$\begin{aligned}\mathsf{ct}_0 &\equiv (u_{\mathsf{ks}}\mathsf{ct}'_1)b + (pe_0^{\mathsf{ks}} + \mathsf{sk})\mathsf{ct}'_1 + \mathsf{ct}'_0\\ &\equiv (u_{\mathsf{ks}}\mathsf{ct}'_1)b + \mathsf{pt} + \left((pe_0^{\mathsf{ks}} + \mathsf{sk})\mathsf{ct}'_1 + pe' - \mathsf{ct}'_1\mathsf{sk}\right)\\ &\equiv (u_{\mathsf{ks}}\mathsf{ct}'_1)b + \mathsf{pt} + \left(pe_0^{\mathsf{ks}}\mathsf{ct}'_1 + pe'\right)\end{aligned}$$

Denote $u = u_{\mathsf{ks}}\mathsf{ct}'_1$, $e_1 = e_1^{\mathsf{ks}}\mathsf{ct}'_1$, and $e_0 = (e_0^{\mathsf{ks}}\mathsf{ct}'_1 + e')$. Then ct can be presented as $(ub + pe_0 + \mathsf{pt}, ua + pe_1)$. Note that the noise e' is bounded due to the correctness of ct'. However, the added noise terms $e_1^{\mathsf{ks}}\mathsf{ct}'_1$ and $e_0^{\mathsf{ks}}\mathsf{ct}'_1$ would be too large, which affects the correctness of decryption. Therefore, we need to reduce

the size of these products to ensure that the decryption of ct' yields the desired plaintext pt. There are two popular techniques for achieving this [BV11,GHS12]. We choose the approach that involves decomposing ct_1 into small coefficients as it maintains the size of the ciphertext modulus. We refer the reader to [KPZ21, Appendix B.1] for more details on how to apply this existing technique.

Regarding to (ii), the nice feature of the transformation described above is that it allows for the homomorphic evaluation of the randomizer u associated with ct to be computed based on the encryption of the randomizer u_{ks}, since $u = u_{\mathsf{ks}} \cdot \mathsf{ct}'_1$. Therefore, it is sufficient to provide $\mathsf{E}(u_{\mathsf{ks}})$, where E is a threshold additively homomorphic encryption scheme. The decryptors then compute $\mathsf{E}(u) = \mathsf{ct}_1 \odot \mathsf{E}(u_{\mathsf{ks}})$, which is the encryption of the randomizer u.

As discussed, the value u is secret. We now mask the encryption of u as follows. During the offline phase, each party $P_i \in \mathcal{P}$ samples $\hat{u}_i \leftarrow \chi_{\mathsf{E}}$ from a noise distribution $\hat{\chi}_{\mathsf{E}}$ which is supported on balls of radii $\hat{r}_{\mathsf{E}}$. The radii $\hat{r}_{\mathsf{E}}$ is chosen to statistically hide u. It then computes the encryption of zero with randomizer $\hat{u}_i$ as $\widehat{\mathsf{ct}}^i = (\hat{u}_i b + pe_0^i, \hat{u}_i a + pe_1^i)$ where $e_0^i, e_1^i \leftarrow \chi_{\mathsf{E}}$, and the encryption of u_i as $\widehat{\mathsf{ct}}_u^i = \mathsf{E}(\mathsf{pk}, u_i)$. Subsequently, it publishes $\widehat{\mathsf{ct}}^i, \widehat{\mathsf{ct}}_u^i$ along with a ZKP $\Pi^{\mathsf{ct}}(\widehat{\mathsf{ct}}^i; \widehat{\mathsf{ct}}_u^i; u_i, e_0^i, e_1^i)$, demonstrating the correctness of $\widehat{\mathsf{ct}}^i, \widehat{\mathsf{ct}}_u^i$ and the well-formedness of u_i. Upon receiving a threshold of $t+1$ such pairs with valid proofs, the parties output the homomorphic sums $\widehat{\mathsf{ct}} = \sum_i \widehat{\mathsf{ct}}_i, \widehat{\mathsf{ct}}_u = \sum_i \widehat{\mathsf{ct}}_u^i$.

Next, during the online phase, the decryptors first apply a key-switch to get ct from ct', and homomorphically evaluate $\mathsf{E}(u)$. Then, they compute $\mathsf{E}(u) \oplus \widehat{\mathsf{ct}}_u$ and use this ciphertext for threshold decryption. After some party successfully combines the decryption shares and obtains $u + \hat{u}$, it can validate it with $\mathsf{ct} \oplus \widehat{\mathsf{ct}}$ and derive the plaintext pt.

Protocol 2 presents our optimized $\mathtt{ThFHE}$ scheme. It leverages a semi-honest decryption, which differs verification of decryption shares which is only executed upon failure, to detect malicious parties a posteriori. We assume that the key switching ks and the encryption of the associated randomizers u_{ks} as $\widehat{\mathsf{ct}}_u = \mathsf{E}(\hat{u})$ are computed during DKG. They can be computed in the same manner as computing the relinearization keys. We denote by Π^{enc} a zk proof of correct encryption, by Π^{zero} a proof of a correct encryption with regards to the message 0, and Π^{ds} a zk proof of correct decryption share generation.

The correctness and security of the optimized protocol is stated in Theorems 4,5 below, and its proof is in Appendix E.8. To simplify the proof we assume TAHE realizes $\mathcal{F}_{\mathsf{TAHE}}$ (Functionality 3). Similar functionalities (up to idiosyncrasies arising in the setup phase) have been proven both for Paillier [FMM+23] and Class-Groups [BDO23] based TAHE. The functionality the optimized protocol achieves is a analogues to $\mathcal{F}_{\mathtt{ThFHE}}$ but uses a randomizer based decryption instead of the regular decryption algorithm, We give the details of the functionality $\mathcal{F}_{\mathtt{ThFHE}}^{\mathsf{preprocess}}$ (Functionality 2) in Appendix E.8.

Theorem 4. *Assume ct' is an encryption of pt with error of size at most B and plaintext modulus p, the key-switching key ks has error of size at most B_{ks} and ciphertext moduli Q. Then threshold decryption with Protocol 2 will output*

PROTOCOL 2 (*Optimized ThFHE Construction*)

1. **Key Generation:**
   (a) P_i has the Shamir secret share sk_i of sk for a TAHE scheme with encryption algorithm E. All parties hold the corresponding public key pk, and the set of verification keys $\{\mathsf{vk}_{i'}\}_{P_{i'}\in\mathcal{P}}$.
   (b) All parties have the public key (a, b) of an $\mathtt{RLWE}$-based FHE scheme, a homomorphic evaluation key evk, as well as a key-switching key ks from the corresponding $\mathtt{RLWE}$ secret key to itself.
   (c) All parties have the corresponding encryption of the randomizers for ks, $\mathsf{ct}_u^{\mathsf{ks}} = \mathsf{E}(\mathsf{pk}, u_{\mathsf{ks}})$.
2. **Encryption**(pt):
   (a) Sample $\tilde{s}, \tilde{e}_0, \tilde{e}_1 \leftarrow \chi_{\mathsf{E}}$.
   (b) Compute $\mathsf{ct}_0 = \mathsf{E}_{\tilde{s}}(\mathsf{pt}, b, p\tilde{e}_0)$ and $\mathsf{ct}_1 = \mathsf{E}_{\tilde{s}}(0, a, p\tilde{e}_1)$ and set $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$.
   (c) Generate a non-interactive ZK proof $\Pi^{\mathsf{enc}}(\mathsf{ct}; \mathsf{pt}, \tilde{s}, \tilde{e}_0, \tilde{e}_1)$ of correct encryption and broadcast $(\mathsf{ct}, \Pi^{\mathsf{enc}})$.
3. **Offline Phase**(B):
   (a) Each party $P_i \in \mathcal{P}$ samples a batch of B randomizers $\hat{u}_i^\tau \leftarrow \hat{\chi}_{\mathsf{E}}$, for $\tau \in [B]$. It computes the encryption of zero as $\widehat{\mathsf{ct}}^{i,\tau} = (\hat{u}_i^\tau b + \hat{e}_{i,0}^{i,\tau}, \hat{u}_i^\tau a + \hat{e}_1^{i,\tau})$ where $\hat{e}_0^{i,\tau}, \hat{e}_1^{i,\tau} \leftarrow \hat{\chi}_{\mathsf{E}}$, and the encryption of $\hat{u}_i^\tau$ as $\widehat{\mathsf{ct}}_u^{i,\tau} = \mathsf{E}(\mathsf{pk}, \hat{u}_i^\tau)$.
   (b) $P_i \in S$ publishes $(\widehat{\mathsf{ct}}^{i,\tau}, \widehat{\mathsf{ct}}_u^{i,\tau})_{\tau\in[B]}$ along with a *B-batched* ZK proof $\Pi_i^{\mathsf{zero}} = \Pi^{\mathsf{zero}}(\widehat{\mathsf{ct}}^{i,\tau}; \widehat{\mathsf{ct}}_u^{i,\tau}; u_i^\tau, e_0^{i,\tau}, e_1^{i,\tau})_{\tau\in[B]}$.
   (c) Upon receiving $(\mathsf{ct}^{\tau,i'}, \widehat{\mathsf{ct}}_{\tau,i'})_{\tau\in[B]}$ along with $, \Pi_{i'}^{\mathsf{zero}}$, $P_i \in S$ verifies the proof. If the proof is invalid, it ignores the message. Otherwise, it records it.
   (d) Upon recording a threshold of $t+1$ such messages, each party sets $\widehat{\mathsf{ct}}^\tau := \sum_{i'} \widehat{\mathsf{ct}}^{i',\tau}$ and $\widehat{\mathsf{ct}}_u^\tau := \sum_{i'} \widehat{\mathsf{ct}}_u^{i',\tau}$, and outputs $(\widehat{\mathsf{ct}}^\tau, \widehat{\mathsf{ct}}_u^\tau)_{\tau\in[B]}$.
4. **Distributed Decryption**(ct', τ):
   (a) Upon receiving a ciphertext $\mathsf{ct}' = (\mathsf{ct}_0', \mathsf{ct}_1')$, each $P_i \in \mathcal{P}$ decomposes ct_1' and performs a key switch, deriving $\mathsf{ct} = (\mathsf{ks}_0\mathsf{ct}_1' + \mathsf{ct}_0', \mathsf{ks}_1\mathsf{ct}_1')$. It also computes $\mathsf{E}(u) = \mathsf{ct}_1' \odot \mathsf{ct}_u^{\mathsf{ks}}$.
   (b) Then, it computes $\mathsf{E}(u) \oplus \widehat{\mathsf{ct}}_u^\tau$, and computes a corresponding decryption share ds_i along with a ZK proof Π_i^{ds}.
   (c) Upon receiving $t+1$ decryption shares, party P_τ reconstruct a plaintext u^*. It uses it to open $\mathsf{ct} + \widehat{\mathsf{ct}}^\tau$.
      – Upon success, it broadcasts u^* to all parties, who will proceed to similarly extract pt.
      – Upon failure, it verifies the ZK proofs Π^{ds} of decryption shares to identify the malicious parties and broadcasts their identities, and awaits for $t+1$ decryption shares with valid proofs.

pt *if* $\frac{Q}{p} > 2^\sigma(B + \log_2(n)B_{\mathsf{ks}})$ *and and the plaintext size of the TAHE scheme is of size at least* $N \cdot 2^n \cdot 2^\sigma Q$.

Theorem 5. *If AHE is semantically secure additively homomorphic encryption and TAHE UC-realizes ideal functionality* $\mathcal{F}_{\mathsf{TAHE}}^{\mathsf{preprocess}}$ *then Protocol 2 UC-realizes the ideal functionality of the decryption in* $\mathcal{F}_{\mathsf{ThFHE}}$ *in the* $\mathcal{F}_{DKG}$*-hybrid model.*

7 Parameter Selection and Efficiency Estimates

In this section, we provide concrete parameter choices and runtime estimates for the optimized variant of our threshold decryption protocol. Our analysis assumes the use of modulus switching before threshold decryption, thereby minimizing the error size. The smallest achievable modulus size q depends on the encryption scheme, primarily influenced by the ℓ_1-norm of the secret key and the expansion factor of the ring. In practical settings, it typically falls within the range of 2^8 to 2^{64}. Another crucial factor affecting the total decryption error is the ring degree, which varies based on noise growth, the computed circuit, and computational security. For simplicity, we fix the computational security level at 128 bits, the statistical security parameter at 40 and impose an upper bound of 2^{13} on the polynomial ring degree.

Plausibly Quantum Secure Construction. In this case, we use our RLWE-based encryption with our denominator-clearing factor as the `ThFHE`. The adjusted modulus, denoted by Q, must accommodate $\log_2(N)$ noise growth from the subset sum, and must be further adjusted according to Theorems 2 and 3. Using this information, we compute the required value of Q that supports threshold decryption given q, n, and estimate the minimal polynomial degree $\hat{n}$ for $\kappa = 128$, based on lattice security estimates [ACC+21]. The highest available Q in these estimations is of size 880 bits. To illustrate how the parameters scale with N, Table 2 presents values for a fixed setting of $\log_2(q) = \log_2(n) = 10$.

Number of Parties	$\log_2(Q)$	$\log_2(\hat{n})$	Ciphertext Size (`KB`)
30	122.13	13	125.05
60	168.63	13	172.67
120	257.63	14	527.62
240	431.63	14	883.97
360	603.97	15	2473.85
480	775.63	15	3176.97

Table 2: Parameter choices and ciphertext size for $\log_2(q) = \log_2(n) = 10$. Original ciphertext size is 1.28`KB`.

In terms of computational complexity, naive implementation requires $O(t^2)$ ring multiplications to compute the Lagrange interpolation coefficients. However, more efficient interpolation algorithms exist (see [BSCKL21]), reducing the complexity to $O(t \cdot \mathsf{polylog}(t))$ ring operations. Using NTT, the complexity of a single ring multiplication is $O(\hat{n}\log(\hat{n})\log^2(Q))$, leading to an overall complexity of $O(t \cdot \mathsf{polylog}(t)\hat{n}\log(\hat{n})\log^2(Q))$. When using a SIMD FHE scheme to encrypt the randomizers, such as our BGV variant, the randomizers from multiple ciphertexts for decryption can be combined upon threshold decryption to compensate for increased ring degree $\hat{n}$.

Additionally, we present the dependency on $\log_2(q)$ and $\log_2(n)$ in Figures 3 and 4. The weak dependence on both parameters suggests that the parameters of the evaluation and threshold decryption schemes are essentially decoupled.

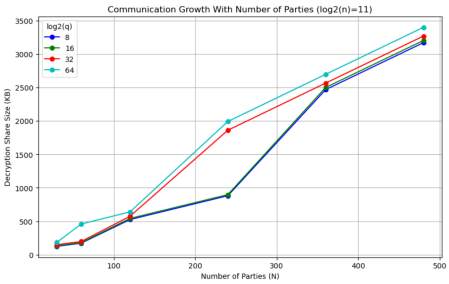


Fig. 3: Fixed degree n.

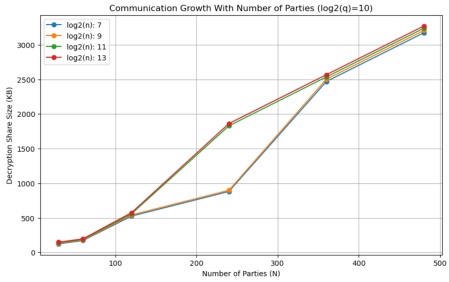


Fig. 4: Fixed modulus q.

Non-Quantum Secure Construction. For our instantiation, we leverage Class Group cryptosystems [BCD+24] due to their transparent setup, though Paillier-based cryptosystems [FMM+23] would yield comparable results. The optimized threshold decryption scheme relies on additive homomorphic operations over integers. As a result, multiple error coefficients can be packed in a single ciphertext, reducing the overhead. The number of ciphertexts for threshold decryption required to encrypt a single FHE randomizer, is approximately equal to the ratio between the number of randomizers and the plaintext modulus. In Class Group-based encryption, the plaintext modulus q_Δ can be fixed. We adopt the setting $\Delta = p_\Delta q_\Delta$ with $p_\Delta \approx q_\Delta$ to balance computational efficiency and communication overhead. Using ciphertext size estimates from [BCIL23], for a security level of $\kappa = 128$, this configuration results in ciphertexts of 4111 bits and plaintext slots of 913.5 bits. The corresponding communication costs are summarized in Figure 5 and are virtually independent of the number of parties.

Non-Quantum Secure Construction. For our instantiation, we leverage Class Group cryptosystems [BCD+24] due to their transparent setup, though Paillier-based cryptosystems [FMM+23] would yield comparable results. The optimized threshold decryption scheme uses additive homomorphic operations over integers, allowing multiple error coefficients to be packed in a single ciphertext and reducing overhead. The number of ciphertexts needed to encrypt a single FHE randomizer is approximately the ratio of randomizers to the plaintext modulus. In Class Group encryption, the plaintext modulus q_Δ can be fixed; we set $\Delta = p_\Delta q_\Delta$ with $p_\Delta \approx q_\Delta$ to balance computational and communication costs. Using estimates from [BCIL23], at security level $\kappa = 128$, this gives ciphertexts of 4111 bits and plaintext slots of 913.5 bits. Communication costs, summarized in Figure 5, are virtually independent of the number of parties.

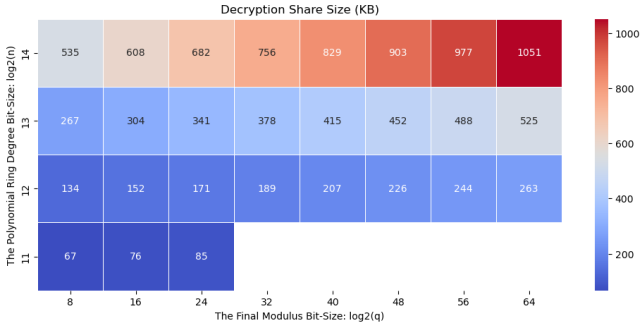


Fig. 5: Heatmap of communication costs using Class Group TAHE.

Regarding computation time, we estimate that for a large number of parties, the reconstruction phase will dominate the overall cost. Based on the estimations in [BCIL23], performing a variable-time exponentiation on a key-sized number (approximately 913.5 bits) takes around 71 ms on a standard laptop (Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz). In the threshold scheme, the exponent for reconstruction scales with N parties, and takes the form $N! \cdot \lambda_{0,j}^S$, which appears to have a bit length of $O(N \cdot \log(N))$. However, this can be optimized to an exponent size of $O(N)$ per decryption share by factoring out common terms, as $|N! \cdot \lambda_{0,j}^S| = \binom{N}{j} \cdot \prod_{i \in [n] \setminus S} |i - j| \cdot \prod_{i \in S} i$. The first factor has size $O(N)$, and the last factor is common across shares. While the middle term appears to have size $(n-t)!$, in practice, many common factors with $\binom{N}{j}$ cancel out, significantly reducing the effective size.

Additionally, we employ multi-exponentiation [Pip80], which allows simultaneous squaring for all bases and restricts multiplications to only the necessary cases. Since the algorithm involves roughly twice as many squarings as multiplications, this yields an additional speedup factor of approximately 3 (depending on the running time of squaring vs multiplying in the group). Consequently, the total computational cost can be estimated as:

$$\underbrace{t}_{\#\text{ Dec Shares}} \cdot \underbrace{\frac{(\log_2(q) + \sigma) \cdot n}{\log_2(q_\Delta)}}_{\#\text{Ciphertexts}} \cdot \underbrace{\frac{N}{3 \log_2(q_\Delta)} \cdot 0.71\,\text{ms}}_{\text{Evaluation Time Per Share}}$$

Since only a single party is required to perform this operation per ciphertext, the cost can be amortized over N ciphertexts. For instance, with $N = 120$, $t = 80$, $\log_2(q) = 16$, and $\log_2(n) = 10$, decryption of 120 ciphertexts takes approximately 26.76 seconds. In throughput, this corresponds to around 4.48 ciphertexts per second. Assuming the original scheme supports batching, we can achieve a throughput of approximately 4587 plaintexts per second.

References

ABGS23. Diego F Aranha, Carsten Baum, Kristian Gjøsteen, and Tjerand Silde. Verifiable mix-nets and distributed decryption for voting from lattice-based assumptions. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1467–1481, 2023.

ABV+12. Shweta Agrawal, Xavier Boyen, Vinod Vaikuntanathan, Panagiotis Voulgaris, and Hoeteck Wee. Functional encryption for threshold functions (or fuzzy IBE) from lattices. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC*, 2012.

ACC+21. Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, et al. Homomorphic encryption standard. *Protecting privacy through homomorphic encryption*, pages 31–62, 2021.

ACD+19. Mark Abspoel, Ronald Cramer, Ivan Damgård, Daniel Escudero, and Chen Yuan. Efficient information-theoretic secure multiparty computation over via galois rings. In *Theory of Cryptography Conference*, pages 471–501. Springer, 2019.

AHI10. Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. *Cryptology ePrint Archive*, 2010.

AJLA+12. Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In *EUROCRYPT*, 2012.

ATZ23. Christoph Aistleitner, Niclas Technau, and Agamemnon Zafeiropoulos. On the order of magnitude of sudler products. *American Journal of Mathematics*, 145(3):721–764, 2023.

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE symposium on security and privacy (SP)*, pages 315–334. IEEE, 2018.

BCCT12. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 326–349, 2012.

BCD+24. Lennart Braun, Guilhem Castagnos, Ivan Damgård, Fabien Laguillaumie, Kelsey Melissaris, Claudio Orlandi, and Ida Tucker. An improved threshold homomorphic cryptosystem based on class groups. *Cryptology ePrint Archive*, 2024.

BCIL23. Cyril Bouvier, Guilhem Castagnos, Laurent Imbert, and Fabien Laguillaumie. I want to ride my bicycl: Bicycl implements cryptography in class groups. *Journal of Cryptology*, 36(3):17, 2023.

BDO23. Lennart Braun, Ivan Damgård, and Claudio Orlandi. Secure multiparty computation from threshold encryption based on class groups. In *Annual International Cryptology Conference*, pages 613–645. Springer, 2023.

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.

BGV14. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully Homomorphic Encryption without Bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):13:1–13:36, 2014.

BS23. Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from lwe with polynomial modulus. Asiacrypt, 2023.

BSCKL21. Eli Ben-Sasson, Dan Carmon, Swastik Kopparty, and David Levit. Elliptic curve fast fourier transform (ecfft) part i: fast polynomial algorithms over all finite fields. *arXiv preprint arXiv:2107.08473*, 2021.

BV11. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, August 2011.

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

CD17. Ignacio Cascudo and Bernardo David. Scrape: Scalable randomness attested by public entities. In *International Conference on Applied Cryptography and Network Security*, pages 537–556. Springer, 2017.

CD24. Ignacio Cascudo and Bernardo David. Publicly verifiable secret sharing over class groups and applications to dkg and yoso. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 216–248. Springer, 2024.

CDC24. Georgiana Crihan, Luminița Dumitriu, and Marian Viorel Crăciun. Preliminary experiments of a real-world authentication mechanism based on facial recognition and fully homomorphic encryption. *Sciences*, 2024.

CGGI16. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2016.

CKKS17. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 409–437. Springer, Heidelberg, December 2017.

CL15. Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from. In *Cryptographers' Track at the RSA Conference*, pages 487–505. Springer, 2015.

CLO+13. Ashish Choudhury, Jake Loftus, Emmanuela Orsini, Arpita Patra, and Nigel P. Smart. Between a rock and a hard place: Interpolating between MPC and FHE. In *ASIACRYPT*, 2013.

CLT18. Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 733–764. Springer, 2018.

CM15. Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO*, 2015.

DDEK+23. Morten Dahl, Daniel Demmler, Sarah El Kazdadi, Arthur Meyre, Jean-Baptiste Orfila, Dragos Rotaru, Nigel P Smart, Samuel Tap, and Michael Walter. Noah's ark: Efficient threshold-fhe using noise flooding. In *Proceedings of the 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 35–46, 2023.

DF91. Yvo Desmedt and Yair Frankel. Shared generation of authenticators and signatures. In *CRYPTO*, 1991.

DJN10. Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. A generalization of paillier's public-key system with applications to electronic voting. *International Journal of Information Security*, 9:371–385, 2010.

DPLS19. Rafaël Del Pino, Vadim Lyubashevsky, and Gregor Seiler. Short discrete log proofs for fhe and ring-lwe ciphertexts. In *PKC*, 2019.

DPSZ12. Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*, pages 643–662. Springer, 2012.

DSDFY94. Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 522–533, 1994.

Feh98. Serge Fehr. Span programs over rings and how to share a secret from a module. Master's thesis, ETH Zurich, 1998.

Fel87. Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *SFCS*. IEEE, 1987.

FMM$^+$23. Offir Friedman, Avichai Marmor, Dolev Mutzari, Yehonatan C Scaly, Yuval Spiizer, and Avishay Yanai. Tiresias: Large scale, maliciously secure threshold paillier. *Cryptology ePrint Archive*, 2023.

FV12. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. ePrint, 2012/144, 2012.

Gen09a. Craig Gentry. A fully homomorphic encryption scheme. In *50th Annual Symposium on Foundations of Computer Science*, 2009.

Gen09b. Craig Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.

GHS12. Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 850–867. Springer, Heidelberg, August 2012.

GO94. Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

Gro09. Jens Groth. Homomorphic trapdoor commitments to group elements. *Cryptology ePrint Archive*, 2009.

HAG$^+$23. Neveen Mohammad Hijazi, Moayad Aloqaily, Mohsen Guizani, Bassem Ouni, and Fakhri Karray. Secure federated learning with fully homomorphic encryption for iot communications. *IEEE IoT Journal*, 2023.

Joy23. Marc Joye. Tfhe public-key encryption revisited. ePrint 2023/603, 2023.

KG23. Dongwoo Kim and Cyril Guyot. Optimized privacy-preserving cnn inference with fully homomorphic encryption. *TIFS*, 2023.

KPZ21. Andrey Kim, Yuriy Polyakov, and Vincent Zucca. Revisiting homomorphic encryption schemes for finite fields. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 608–639. Springer, Heidelberg, December 2021.

LATV12. Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *ACM symposium on Theory of computing*, 2012.

LPR13. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In *EUROCRYPT*, 2013.

LPSY19. Yehuda Lindell, Benny Pinkas, Nigel P Smart, and Avishay Yanai. Efficient constant-round multi-party computation combining bmr and spdz. *Journal of Cryptology*, 32:1026–1069, 2019.

Mar93. Tom Marley. Graded rings and modules. *Some notes based on a five-week course*, 1993.

MBH23. Christian Mouchet, Elliott Bertrand, and Jean-Pierre Hubaux. An efficient threshold access-structure for rlwe-based multiparty homomorphic encryption. *J. Cryptol.*, 36(2):10, 2023.

MTPBH21. Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. Multiparty homomorphic encryption from ring-learning-with-errors. PoPETs, 2021.

MW16. Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In *EUROCRYPT*, 2016.

OSV20. Emmanuela Orsini, Nigel P Smart, and Frederik Vercauteren. Overdrive2k: efficient secure mpc over from somewhat homomorphic encryption. In *RSA*, 2020.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology-EUROCRYPT 99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2-6, 1999 Proceedings 18*, pages 223–238. Springer, 1999.

Pip80. Nicholas Pippenger. On the evaluation of powers and monomials. *SIAM Journal on Computing*, 9(2):230–250, 1980.

RAD78. R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 1978.

RRJ+22. Tim Ruffing, Viktoria Ronge, Elliott Jin, Jonas Schneider-Bensch, and Dominique Schröder. Roast: robust asynchronous schnorr threshold signatures. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2551–2564, 2022.

Sch99. Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Annual International Cryptology Conference*, pages 148–164. Springer, 1999.

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

Sho00. Victor Shoup. Practical threshold signatures. In *EUROCRYPT*, 2000.

Wri64. EM Wright. Proof of a conjecture of sudler's. *The Quarterly Journal of Mathematics*, 15(1):11–15, 1964.

A Standard Definitions

For completeness, some standard definitions in the literature are provided in this section.

Definition 5 ([Gro09]). *A non-interactive commitment scheme consists of a pair of PPT algorithms* $(\mathbf{setup}, \mathbf{com})$. *The setup algorithm* $\mathsf{pp} \leftarrow \mathbf{setup}(1^\kappa, \mathsf{aux})$, *where* κ *is a security parameter and* aux *is some auxiliary input, outputs the public parameters for the commitment scheme. The public parameters* pp *also determine the message space* $\mathcal{MS}_{\mathsf{pp}}$, *randomness space* $\mathcal{RS}_{\mathsf{pp}}$ *and commitment space* $\mathcal{CS}_{\mathsf{pp}}$. *The commitment algorithm* $\mathbf{com}_{\mathsf{pp}}$ *defines a function* $\mathcal{MS}_{\mathsf{pp}} \times \mathcal{RS}_{\mathsf{pp}} \to \mathcal{CS}_{\mathsf{pp}}$. *For a message* $x \in \mathcal{MS}_{\mathsf{pp}}$, *the algorithm draws* $r \leftarrow \chi_{\mathsf{pp}}$ *from a distribution supported on* $\mathcal{RS}_{\mathsf{pp}}$, *and computes a commitment* $C = \mathbf{com}_{\mathsf{pp}}(x; r)$. *Whenever the public parameters are clear from the context we write* $\mathbf{com}$ *instead of* $\mathbf{com}_{\mathsf{pp}}$.

A commitment scheme usually has hiding *and* binding *properties as follows:*

- Computational hiding. *For every PPT adversary* $\mathcal{A}$, *every* $\mathsf{pp} \leftarrow \mathbf{setup}(1^\kappa, \mathsf{aux})$ *and every* $x_0, x_1 \in \mathcal{MS}_{\mathsf{pp}}$:

$$\left| \Pr[\mathcal{A}(x_0, x_1, \mathbf{com}(x_b; r)) = b] - \frac{1}{2} \right| \leq \mathrm{negl}(\kappa),$$

  where the probability is under $b \to \{0,1\}$, $r \leftarrow \chi_{\mathsf{pp}}$ *and uniformly random coins of* $\mathcal{A}$.
- Computational binding. *For every PPT adversary* $\mathcal{A}$:

$$\Pr\begin{bmatrix} (x_0, x_1, r_0, r_1) \leftarrow \mathcal{A}(\mathbf{setup}(1^\kappa, \mathsf{aux})) \\ \mathbf{com}(x_0; r_0) = \mathbf{com}(x_1; r_1) \wedge x_0 \neq x_1 \end{bmatrix} \leq \mathrm{negl}(\kappa)$$

  where the probability is under the random coins of $\mathcal{A}$ *and* $\mathbf{setup}$. *The above assumes* $x_0, x_1 \in \mathcal{MS}_{\mathsf{pp}}$ *and* $r_0, r_1 \in \mathcal{RS}_{\mathsf{pp}}$.

Definition 6. *([CD17]) IND1-secrecy: We say that a PVSS is IND1-secret if for any polynomial time adversary* $\mathcal{A}_{Priv}$ *corrupting at most* $t-1$ *parties,* $\mathcal{A}_{Priv}$ *has negligible advantage in the following game played against a challenger:*

1. *The challenger runs* $\mathbf{setup}$ *as the dealer and sends the public information to the adversary.*
2. $\mathcal{A}_{Priv}$ *creates secret keys for the corrupted parties and send the corresponding public keys to the challenger.*
3. *The challenger chooses values* x_0, x_1 *at random in the space of secrets. Furthermore it chooses* $b \leftarrow \{0,1\}$ *uniformly at random. It runs* $\mathsf{Dist}(x_b)$ *and sends all public information to* $\mathcal{A}$.
4. $\mathcal{A}_{Priv}$ *outputs* $b' \in \{0,1\}$

PARAMETERS: The noise distribution χ_e, The key distribution χ_s, the plaintext space $\mathcal{R}_P$, the key space $\mathcal{R}_Q$ and $\beta = \lfloor \log_2(Q) \rfloor$.

$\underline{\mathbf{keygen}(1^\kappa) \to (\mathsf{sk}, \mathsf{pk}, \mathsf{lk})}$
- Sample $s \leftarrow \chi_s$
- Sample $a, a_{i\in[\beta]} \leftarrow \mathcal{R}_Q$ and $e, e_{i\in[\beta]} \leftarrow \chi_e$
- Output $(\mathsf{sk}, \mathsf{pk}, \mathsf{lk})$ where $\mathsf{sk} = s$, $\mathsf{pk} = (a, [-a \cdot s + e]_Q)$, and $\mathsf{lk} = \{\mathsf{lk}_1, \ldots, \mathsf{lk}_\beta\}$ where $\mathsf{lk}_i = (a_i, [-a_i \cdot s + e_i + 2^i s^2]_Q)$ for (a_i, e_i)

$\underline{\mathsf{E}(\mathsf{pk}, m) \to \mathsf{ct}}$
- Sample $u \leftarrow \chi_s$ and $e_1, e_2 \leftarrow \chi_e$
- Let $p_0 = \mathsf{pk}[0]$, $p_1 = \mathsf{pk}[1]$
- Compute $c_0 = [p_0 \cdot u + e_1 + \lfloor \frac{Q}{P} \rfloor \cdot m]_Q$ and $c_1 = [p_1 \cdot u + e_2]_Q$
- Output (c_0, c_1)

$\underline{\mathsf{D}(\mathsf{sk}, \mathsf{ct}) \to m}$
- Let $c_0 = \mathsf{ct}[0]$, $c_1 = \mathsf{ct}[1]$
- Compute $v = [c_0 + c_1 \cdot s]_Q$
- Output $[\frac{1}{\lfloor \frac{Q}{P} \rfloor} \cdot v]_P$

$\underline{\mathsf{EV}(\mathsf{lk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k) \to \mathsf{ct}}$
- Output $\mathsf{E}(\mathsf{pk}, C(m_1, \ldots, m_k))$ where m_i is the plaintext of ct_i. Note that the multiplication involves reducing the size of the ciphertext (re-linearization), which is achieved by using the lk.

Fig. 6: BGV-based HE Construction. The fully HE scheme enables the computation of addition and multiplication on ciphertexts. Since any function can be expressed as a combination of XOR and AND gates, HE allows for the evaluation of arbitrary functions on encrypted data. Thus, we use $\mathsf{EV}()$ to present the homomorphic evaluation on any circuit C.

Definition 7 (RLWE). *Let $\Phi_{2n}(x) = x^n + 1$ be the power-of-two cyclotomic polynomial of degree n, let $\mathcal{R} = \mathbb{Z}[x]/\langle \Phi_{2n}(x) \rangle$ be the corresponding cyclotomic ring and let $q \in \mathbb{N}$ be a prime, denoting $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle \Phi_{2n}(x) \rangle$. Let $\chi_{\mathsf{E}}, \chi_{\mathsf{s}}$ be distributions over $\mathcal{R}_q$.*

For a ring element $\mathsf{s} \in \mathcal{R}$, consider the distribution A_{s} over $\mathcal{R}^2$ defined as $\{(a, \mathsf{s}a + e \mod q)\}$, where $a \leftarrow \mathcal{U}(\mathcal{R}_q)$ is sampled uniformly, e is sampled from χ_{E}. Then the (decisional) RLWE problem with respect to parameters $(n, q, \chi_{\mathsf{s}}, \chi_{\mathsf{E}})$ is to distinguish A_{s} from the uniform distribution $\mathcal{U}(\mathcal{R}_q^2)$, given an a-priori unbounded number of samples, where s is drawn from χ_{s}.

B PVSS over Polynomial Rings

In this section, we turn the secret sharing scheme over $\mathcal{R}_Q$ above to be verifiable. Looking ahead, we will use the verifiability property in order to derive verification keys for the parties during DKG, which can be used to prove correctness of the decryption shares during threshold decryption. Essentially, verification keys can be thought of as a commitments to the individual parties secret shares. At the

same time, they bind the parties to use their shares upon threshold decryption, as well as hide the underlying shares. Moreover, using a homomorphic encryption scheme also assists with distributed generation and validation of the verification keys, as shown in Appendix C.

Importantly, in our instantiation, secret shares sk_i need not be small, and only the secret key sk has a bounded norm. Therefore, taking an encryption of zero $\mathbf{com}'(\mathsf{sk}_i; e_i) = \mathsf{E}_{\mathsf{sk}_i}(0, a, e_i)$ with bounded error as a verification key is not binding. Namely, for any given a and a given value of the commitment, the adversary can compute for each e_i an sk'_i that will satisfy the relation. To this end, we offer a commitment of the form $\mathbf{com}(\mathsf{sk}_i; e_i, e'_i) := (\mathsf{E}_{\mathsf{sk}_i}(0, a, e_i), \mathsf{E}_{\mathsf{sk}_i}(0, a', e'_i)) = (a\mathsf{sk}_i + e_i, a', a'\mathsf{sk}_i + e'_i)$, where e_i and e'_i are small. Here, a and a' are public parameters of the commitment scheme, which can be shared among all parties, and in particular can be used for multiple commitments.

Formally, we consider the following commitment scheme $\mathsf{COM}_{\mathsf{async}} = (\mathbf{setup}, \mathbf{com})$. First, $\mathbf{setup}(1^\kappa, 1^\sigma, \mathsf{aux})$ outputs the ciphertext power-of-two ring degree n, a prime modulus Q such that Φ_{2n} is irreducible mod Q, and noise distribution χ_{E} with support r_{E} such that the $\mathtt{RLWE}$ problem over $\mathcal{R}_Q$ is κ-computationally hard. It then sets $\mathcal{MS} = \mathcal{R}_Q$ as the message space, $\mathcal{RS} = \{e \in \mathcal{R} : \|e\|_{\infty,\Phi} \leq \max(r_{\mathsf{E}}, \sqrt{Q} \cdot 2^{-1-\sigma/2n})\}$ as the randomness space, and $\mathcal{CS} = \mathcal{R}_Q^2$ as the commitment space. Then, it samples $(a, a') \leftarrow \mathcal{R}_Q^2$ from the random oracle $\mathcal{O}$. The commitment $\mathbf{com} : \mathcal{MS} \times \mathcal{RS} \mapsto \mathcal{CS}$ is then defined as $\mathbf{com}(s; (e, e')) = (as + e, a's + e')$. Note that this alone will not actually be a commitment and must be accompanied by a proof of of encryption with a secret s and small errors e, e' denoted $\Pi^{\mathbf{com}}$. Its security is stated below:

Lemma 5. *If $\Pi^{\mathbf{com}}$ is computationally zero-knowledge and statistically (or computationally) sound $\mathsf{COM}_{\mathsf{async}}$ is computationally hiding and statistically (or computationally) binding.*

Proof. Computational hiding follows directly from the $\mathtt{RLWE}$ assumption and the zero-knowledge property of the zk-proof. As for statistically binding, we bound the probability that there exist two solutions s_1, e_1, e'_1 and s_2, e_2, e'_2 such that $\mathbf{com}(s_1; e_1, e'_1) = \mathbf{com}(s_2; e_2, e'_2)$ and $s_1 \neq s_2$. Since Q is prime and Φ_{2n} is chosen irreducible mod Q, $\mathcal{R}_Q$ is a field. Therefore, these equations imply that $a = \frac{(s_1 - s_2)}{(e_1 - e'_1)}$ and $a' = \frac{(s_1 - s_2)}{(e_2 - e'_2)}$, which implies $\frac{a}{a'} = \frac{(e_1 - e'_1)}{(e_2 - e'_2)}$. Since a, a' are random, a/a' is uniform over $\mathcal{R}_Q$, a set of $|\mathcal{R}_Q| = Q^n$ ring elements. However, $\|e_1 - e'_1\|_{\infty,\Phi}, \|e_2 - e'_2\|_{\infty,\Phi} \leq \sqrt{Q} \cdot 2^{-\sigma/2n}$, and therefore, $e_1 - e'_1$ and $e_2 - e'_2$ can take up to $Q^{n/2} \cdot 2^{-\sigma/2}$ values, and so the ratio takes up to $Q^n \cdot 2^{-\sigma}$ values. Therefore by union bound, the chance that there exist such solutions is at most $2^{-\sigma}$, which is statistically negligible. From the soundness of the zk-proof the sender of the commitment must use errors bounded as above. □

Building on $\mathsf{COM}_{\mathsf{async}}$, Protocol 3 establishes PVSS. We first prove that it is a VSS via proofs of fairness and t-zero-knowledge are provided in Appendix E.3. As all proofs in the protocol are publicly verifiable the overall protocol is a PVSS similar to the simple optimized approach [Sch99].

PROTOCOL 3 (*PVSS for RLWE -* $\Pi_{\mathsf{PVSS}}(s)$)

1. **Dist**(s):
   - The dealer $\mathcal{D}$ samples $e, e' \leftarrow \chi_{\mathsf{E}}$. Then, it samples t coefficients $f_1, \ldots, f_t \leftarrow \mathcal{R}_Q$ and calculates the secret share $s_i = f(\alpha_i) = s + \sum_{k=1}^{t} f_k \alpha_i^k$ of each party P_i, associated with the interpolation point $\alpha_i \in \mathcal{I}$.
   - It then computes $F_k = \mathsf{COM}_{\mathsf{async}}(f_k)$ for $k \in [t]$ as well as $F_0 = \mathsf{COM}_{\mathsf{async}}(f_0)$.
   - It also computes $\mathsf{ct}_i = \mathsf{E}_{\mathsf{pk}_i}(s_i)$, an encryption of the share $s_i = f(\alpha_i)$ of each party.
   - It also computes the verification key of each party as $\mathsf{vk}_i = \sum_k F_k \alpha_i^k$, .
   - It then broadcasts $\{F_k\}_{k\in[0,t]}$ and $\{\mathsf{ct}_i\}_{i\in[N]}$ along with a ZK proof $\Pi^{\mathsf{PVSS}}(\{F_k\}_{k\in[0,t]}, \{\mathsf{ct}_i\}_{i\in[N]}; s, \{f_k\}_{k\in[t]})$ which includes the proofs $\Pi_k^{\mathbf{com}}$ for each commitment F_k along with proofs $\Pi_i^{\mathbf{com},\mathsf{enc}}(\mathsf{ct}_i, \mathsf{vk}_i; s_i)$ that ct_i is an encryption of $s_i = f(\alpha_i)$, committed under vk_i.
2. **VerifyDist**: Each party $P_i \in \mathcal{P}$ then
   - Verifies the proofs $\Pi_k^{\mathbf{com}}$.
   - Computes the verification key of each party $\mathsf{vk}_j = \sum_k F_k \alpha_j^k$ and verifies the proofs $\Pi_i^{\mathbf{com},\mathsf{enc}}$. If any verification fails, ignore the message.
   - Decrypts its share $s_i = \mathsf{D}_{\mathsf{sk}_i}(\mathsf{ct}_i)$.

C Asynchronous DKG

In this section, we present three DKG protocols. The first one follows a straightforward approach, and is used for two purposes. First, it captures most of the technicalities with regards to the structure of the evaluation key, and recalls the techniques of gadget decomposition. Second, it demonstrates the challenginess of designing a protocol over an asynchronous broadcast consensus channel, wherein the resulting public parameters for the FHE scheme are weakly affected by N. It is presented in Section C.1. Then, Section C.2 realizes the key generation for Protocol 1, our first `ThFHE` construction from Section 5. Lastly, Section C.3 realizes the key generation for Protocol 2, our optimized `ThFHE` construction from Section 6 that leverages preprocessing techniques.

C.1 First Approach

As stated in Protocol 1, the DKG should generate the following output:

1. A secret share sk_P of the secret `ThFHE` decryption key sk for each party.
2. The corresponding public key $\mathsf{pk} = (a, b)$ where $a \in \mathcal{R}_Q$, and $b = a\mathsf{sk} + \Delta e \in \mathcal{R}_Q$ and e is "small".
3. The set of verification keys $\{\mathsf{vk}_{P'}\}$ of all parties, where $\mathsf{vk}_{P'} = \mathbf{com}(\mathsf{sk}_{P'})$ are homomorphic commitments over the corresponding key shares, where **com** is statistically hiding and computationally binding. Specifically, **com** can be instantiated with $\mathsf{COM}_{\mathsf{async}}$ from Section B.

4. Additionally, the protocol outputs a relinearization key lk, which is essentially a key switching key from sk^2 to sk, and can be thought of as an encryption of a decomposition of sk^2 under sk. For ease of exposition, we do not describe generation of the bootstrapping key which takes a similar form.

We use a *powers of* w gadget decomposition for the relinearization key lk. The gadget vector $\mathbf{g}_{w,\ell_0} = (Qw^{-1}, \ldots, Qw^{L-\ell_0}) \in \mathcal{R}_Q^{\ell}$ is a row vector of ring elements. The corresponding "decomposition operator" $\mathbf{g}_{w,\ell_0}^{-1}(s) : \mathcal{R} \to \mathcal{R}^{\ell}$ returns the decomposition of the fraction $s/Q \in \mathbb{Q}[x]/\Phi_n(x)$ in base w, taking the first ℓ_0 digits. Formally, if $s/Q = \sum_{\ell=1}^{\infty} s_\ell Q w^{-\ell}$ where for each ℓ $\|s_\ell\|_\infty < \lfloor w/2 \rfloor$, $\mathbf{g}_{w,\ell}^{-1}(s) = (s_1, \ldots, s_{\ell_0})$. Note that $\left\|\mathbf{g}_{w,\ell_0}^{-1}(s)\right\|_\infty \leq w/2$. In addition, $\left\|\mathbf{g}_{w,\ell_0} \circ \mathbf{g}_{w,\ell_0}^{-1}(s) - s\right\|_\infty \leq Qw^{-\ell_0}/2$. Essentially, both $w/2$ and $Qw^{-\ell_0}/2$ must be small for correctness of key-switching. Nonetheless, we use this gadget decomposition technique since from the resulting key-switching and relinearization keys for the top level of the moduli ladder, the keys for all the other levels are derived by a modulo-switch (or rescaling in CKKS) operation that is performed locally.

As we have seen in Section 5, the ciphertext moduli Q is adjusted exponentially in order to allow for threshold decryption. This stems from the fact that the Lagrange coefficients are multiplied by the encryption noise terms that must be kept small. A similar issue arises during DKG. Specifically, when computing the relinearization key $\mathsf{lk}_\ell = \mathsf{E}_{\mathsf{sk}} g_\ell \mathsf{sk}^2, a_\ell, e_\ell^{\mathsf{lk}}$. A naive approach would involve computing the key-switching key $\mathsf{ks}_\ell = \mathsf{E}_{\mathsf{sk}} g_\ell \mathsf{sk}, a_\ell, e_\ell^{\mathsf{ks}}$, then let each party multiply by its secret share sk_i, and finally interpolate $t+1$ shares to effectively multiply by sk. However, this runs into the aforementioned issue, namely, the encryption error overflows Q. One approach to resolve this could be to start from $\mathsf{ks}_\ell = \mathsf{E}_{\mathsf{sk}} g_\ell \mathsf{sk}, a_\ell, \Delta e_\ell^{\mathsf{ks}}$ and use Corollary 1 to bound the noise of the interpolated relinearization key.

In Protocol 4, we follow the above approach. Namely, encryption noise is multiplied by Δ, and in particular, the public parameters of the underlying FHE scheme for homomorphic circuit evaluation will depend on the number of parties N. As a result, it cannot be used for our optimized construction, where we only allow the size of the ciphertext for decryption to scale with N. Therefore, we propose in the next section another DKG protocol that avoids this issue, and then augment to fully support the optimized construction.

Protocol. The key generation aims to produce the secret key sk, the public key pk, and the relinearization key lk, with each key following a specific formula depending on the FHE scheme used. For simplicity, we assume the secret key sk is randomly sampled from the same domain as the encryption error χ_{E}, which is a common choice. The public key pk is generally expressed as $(a, a\mathsf{sk}+e)$, where a is a constant and e is noise. The relinearization key $\mathsf{lk}_\ell = (a_\ell, \mathsf{E}_{\mathsf{sk}}(\mathbf{g}[\ell]\mathsf{sk}^2, a_\ell, \Delta e_\ell)$ encrypts a decomposition of sk^2. A tensor product of two ciphertext encrypted under sk, results with a ciphertext encrypted under $(\mathsf{sk}, \mathsf{sk}^2)$. Given lk, a key-switch back to sk is enabled. Protocol 5 presents our construction for generating

PROTOCOL 4 (*DKG: First Attempt*)

Setup: Each party $P_i \in \mathcal{P}$ retrieves $a, \{a_\ell\}_{\ell\in[0,\ell_0]} \leftarrow \mathcal{R}_Q$ from the Random Oracle $\mathcal{O}$. In addition, each party samples a public-private key pair and computes a corresponding ZKP for the underlying PVSS scheme.

1. **Round 1**: Each party $P_i \in \mathcal{P}$
   (a) Samples $s_i, e_i, \{e_{i,\ell}\}_{\ell\in[0,\ell_0]} \leftarrow \chi_{\mathsf{E}}$.
   (b) Calculates $b_i = \mathsf{E}_{s_i}(0, a, \Delta e_i)$ and $b_i^\ell = \mathsf{E}_{s_i}(w^{L-\ell}s_i, a_\ell, \Delta e_{i,\ell})$ for $\ell \in [0, \ell_0]$.
   (c) Generates a ZK proof $\Pi_i(b_i, \{b_i^\ell\}_{\ell\in[0,\ell_0]}; s_i, e_i, \{e_{i,\ell}\}_{\ell\in[0,\ell_0]})$ of correct encryption of s_i and its powers of w decomposition.
   (d) Calls protocol $\Pi_{\mathsf{PVSS}}.\mathsf{Dist}(s_i)$ to verifiably share its secret s_i.
   (e) Broadcasts $(b_i, \{b_i^\ell\}_{\ell\in[0,\ell_0]}, \Pi_i, \{v_{i,j}\}_{j\in[N]})$ to all parties.
   (f) Generates and broadcasts a proof $\Pi_i'(b_i, F_{i,0}; s_i)$ that binds b_i to the committed s_i of the PVSS.
2. **Round 2**: Upon receiving a subset $S_1 \subset \mathcal{P}$ of $t+1$ valid messages with verified proofs and verified $\Pi_{\mathsf{PVSS}}.\mathsf{VerifyDist}()$, each party $P_i \in \mathcal{P}$
   (a) Computes $b = \sum_{j\in S_1} b_j$, $b^\ell = \sum_{j\in S_1} b_j^\ell$ for $\ell \in [0, \ell_0]$, $\mathsf{sk}_i = \sum_{j\in S_1} s_{j,i}$ and $\mathsf{vk}_{j'} = \sum_{j\in S_1} v_{j,j'}$ for each $j' \in [N]$. Denote $\mathsf{pk} = (a, b)$.
   (b) Samples $\{u_{i,\ell}, e'_{i,\ell}, e''_{i,\ell}\}_{\ell\in[0,\ell_0]} \leftarrow \chi_{\mathsf{E}}$.
   (c) Sets $a_{i,\ell}^0 = \Delta u_{i,\ell} \cdot a + \Delta e'_{i,\ell}$ and $b_{i,\ell}^0 = 0 + \Delta u_{i,\ell} \cdot b + \Delta e''_{i,\ell}$.
   (d) Computes $(a_{\mathsf{lk},i}^\ell, b_{\mathsf{lk},i}^\ell) = \mathsf{sk}_i \cdot (a_\ell, b^\ell) + (a_{i,\ell}^0, b_{i,\ell}^0)$.
   (e) Generates a corresponding ZK proof $\Pi_i^{\mathsf{lk}}(\{(a_{\mathsf{lk},i}^\ell, b_{\mathsf{lk},i}^\ell)\}_{\ell\in[0,\ell_0]}, \mathsf{vk}_i; \mathsf{sk}_i, \{(u_{i,\ell}, e''_{i,\ell}, e'_{i,\ell})\}_{\ell\in[0,\ell_0]})$.
   (f) Broadcasts $(\{(a_{\mathsf{lk},i}^\ell, b_{\mathsf{lk},i}^\ell)\}_{\ell\in[0,\ell_0]}, \Pi_i^{\mathsf{lk}})$.

   Output: Upon receiving a subset $S_2 \subset \mathcal{P}$ of $t+1$ valid messages with verified proofs, each party $P_i \in \mathcal{P}$
   (a) Calculate $\mathsf{lk}_\ell = \sum_{j\in S_2} \lambda_{j,0}^{S_2}(a_{\mathsf{lk},j}^\ell, b_{\mathsf{lk},j}^\ell)$ for each $\ell \in [0, \ell_0]$.
   (b) Output $(\mathsf{pk}, \mathsf{lk}, \{\mathsf{vk}_j\}_{j\in[N]}; \mathsf{sk}_i)$.

these keys in a distributed setting, when the adversary controls up to t of the parties.

The protocol is implemented over an asynchronous consensus channel, implemented over an asynchronous reliable broadcast channel. In particular, all honest parties retrieve all messages in the same order, and therefore, there is agreement on the set S_i of the first $t+1$ valid messages received in each round.

At a high level, our approach follows a $(t+1)$-out-of-$(t+1)$ key generation protocol, as described in [AJLA+12], where a subset S of $t+1$ parties is involved, with each party selecting a random secret s_i. Each party P_i acts as a dealer, distributing its additive share s_i among all N parties using the PVSS protocol from Section B. These additive shares are then combined by each party in order to compute its secret share of $\mathsf{sk} := \sum_S s_i$. By using a PVSS scheme, the distributed shares and ensured to be correct, and also, a homomorphic commitment on the underlying secret shares vk_i can be homomorphically evaluated and validated for each party.

In addition to sending a PVSS over s_i, each party also sends $b_i = \mathsf{E}_{s_i}(0, a, \Delta e_i)$ from which the public key $b = \sum_S b_i$ can be derived. It is tied to the underlying secret sk that is shared with PVSS by a proper ZKP.

To compute the relinearization key $\{\mathsf{lk}_\ell\}_{\ell\in[0,\ell_0]}$, each party P_i broadcasts in addition to b_i, the terms $b_i^\ell = \mathsf{E}_{s_i}(\mathbf{g}[\ell]s_i, a_\ell, \Delta e_{i,\ell})$. Summing over all parties in S_1, the parties compute $b^\ell = \sum_S b_i^\ell$ which is a key-switch from sk to itself. Then, in the second round, each party multiplies b^ℓ by its secret share sk_i, and homomorphically masks it with an encryption of zero (a^0, b^0). After receiving $t+1$ such shares, they can be combined by a Lagrange interpolation to get lk_ℓ, the relineaization key. Here, it is important that all noise terms are multiplied by the denominator clearing factor Δ, as otherwise, the noise term in lk will not have a bounded error.

In Appendix E.4, we provide a concrete analysis of the noise growth during the DKG of the above Protocol 4. Security analysis is only provided for the following protocols which result with a significantly reduced noise growth.

Theorem 6. *Assume $\mathcal{A}$ adaptively blocks up to $f \le N/3$ parties in each communication round, and statically corrupts up to $t < N$ parties. Then Protocol 4 outputs a well-formed tuple $(\mathsf{pk}, \mathsf{lk}, \{\mathsf{vk}_j\}_{j\in[N]}; [\mathsf{sk}])$ wherein:*

- *The public key $\mathsf{pk} = (a, b)$ where $b = a\mathsf{sk} + \Delta e$ corresponds to the shared secret key $[\mathsf{sk}]$.*
- $\|\mathsf{sk}\|_{\infty,\Phi(n)}, \|e\|_{\infty,\Phi(n)} \le t + 1r_{\mathsf{E}}$.
- *The corresponding verification keys $\mathsf{vk}_i = \mathbf{com}(\mathsf{sk}_i)$ are well-formed.*
- *The relinearization keys $\mathsf{lk}_\ell = (a_\ell, b^\ell)$ are well-formed, namely $b^\ell = a_\ell\mathsf{sk} + g_{w,\ell_0}[\ell]\mathsf{sk}^2 + e_\ell^{\mathsf{lk}}$.*
- $\|\mathsf{lk}\|_{\infty,\Phi(n)} \le 2(t+1)^2 r_{\mathsf{E}}^2\gamma\|\Delta\|_{\infty,\Phi_n} + (t+1)r_{\mathsf{E}}^2(\gamma+1)\left\|\Delta\cdot\lambda^S_{(\cdot),0}\right\|_{\infty,\Phi_n} = \mathit{exp}(N)$, *wherein γ is the expansion factor of $\mathcal{R}$ ($\gamma = n$ for powers of two cyclotomic rings).*

We refer to Corollary 1 for theoretical bounds on $\|\Delta\|_{\infty,\Phi_n}$ and $\left\|\Delta\cdot\lambda^S_{(\cdot),0}\right\|_{\infty,\Phi_n}$, and Appendix D for a tighter experimental analysis. We note that in any case, this terms grow exponentially with the number of parties N. This is addressed in the following section.

While the above theorem considers a malicious adversary, it does not ensure zero-knowledge. We stress that we do not include a UC simulation for this protocol. This is primarily because we find it hard to UC-simulate the relinearization key, in the asynchronous framework. However, we do believe the protocol to be secure, as the ZKPs enforce the adversary to be semi-honest, and the transcript of the protocol only involves ZKPs and ciphertexts of a CPA-secure encryption scheme. Nevertheless, the DKG protocols that we propose to use in our constructions (Protocols 1,2) are proven secure with a UC-simulation.

C.2 DKG of Protocol 1

Nevertheless, the above approach results with an overwhelming factor exponential with N for the noise of the relinearization key. As a result, the public

parameters for the underlying FHE scheme become prohibitively large, since the ciphertext moduli should be increased accordingly, and then the ring degree should also be adjusted to maintain security. In this section we resolve this issue, and then in the next Section C.3 we extend the protocol to support our optimized threshold decryption, as described in Section 6.

Essentially, our idea is to first describe a protocol wherein the set S_1 of $t+1$ parties who participated in the first communication round of the DKG, is effectively online for the second round as well. This way, the subset of parties S_2 participating in the second round is effectively pre-determined ($S_2 = S_1$), which allows the parties to know the Lagrange coefficients in advance. Therefore, they can derive locally an *additive* sharing over sk, namely, $\{\lambda_{i,0}^{S_1}\mathsf{sk}_i\}_{i\in S_1}$. Each party then multiplies ks by its additive share $\lambda_{i,0}^{S_1}\mathsf{sk}_i$, and summing up results with a well-formed relinearization key with a small $t+1$ factor overhead on the noise term.

ROAST. Specifically, we use a trick proposed by ROAST [RRJ+22], a wrapper around FROST, a threshold Schnorr protocol, that provides robustness and asynchronity. Specifically, their protocol consists of two communication rounds and local output: (i) a presign round, wherein a *post-determined* subset S_1 of $t+1$ parties send nonces r_i; (ii) an online round wherein effectively the same subset of $t+1$ parties send there partial signature z_i of the form $z_i = a_i + \lambda_{i,0}^{S_1} \cdot b_i$; and (iii) combining the partial signatures which involves computing $z = \sum_{i\in S_1} z_i$. In a nutshell, by predicting the set $S_2 = S_1$ of parties that will partial sigantures, combining the shares can be done more efficiently, and only consists of additions.

We observe that the DKG Protocol 4 already takes a similar form, where in the first round a post-determined subset of parties S_1 samples the public key $b = \sum_{i\in S_1} b_i$, but then in the second round, we want parties to expect a pre-determined subset S_2 of parties and multiply the ciphertext $(a_{\mathsf{lk}}, b_{\mathsf{lk}})$ by $\lambda_{i,0}^{S_2} \cdot \mathsf{sk}_i$, so that in the output phase summing over S_2 will result with a homomorphic multiplication by sk which is bounded overall. We may therefore apply the exact same transformation, which we briefly cover below.

Essentially, ROAST translate the execution with pre-determined subset of participants, into execution of multiple sessions of the same protocol that run concurrently with the same subset of participants. Specifically, session $\mathsf{sid}+1$ may run in parallel with session sid. ROAST keeps track of two sets: R_{sid} is the set of responsive parties, and M is the set of known malicious signers. Once a party becomes a member of M, all messages from that party will be ignored.

The ROAST protocol begins with session $\mathsf{sid} = 0$ wherein parties are requested to send their nonces, and M and every R_{sid} are initially the null set. Upon receiving a request to begin the protocol with $\mathsf{sid} = 0$, the parties respond with their first round message $r_0^{(}0)$ only; this will be different for every session $\mathsf{sid} > 0$. After receiving the responses for a given session id sid, they are placed into R_{sid}. Once $|R_{\mathsf{sid}}| = t+1$ messages are received, parties from R_{sid} respond with their second round message $z_i^{(\mathsf{sid})}(R_{\mathsf{sid}})$, along with a first round message nonce $r_i^{(\mathsf{sid}+1)}$ for the next session. Essentially, parties "piggyback" the next session's

nonce with the current session's signature share. Importantly, $z_i(R_{\mathsf{sid}})$ means that the parties expect to retrieve shares $z_i^{(\mathsf{sid})}$ from each party in R_{sid}. However, to avoid starvation, another protocol is instantiated in parallel, in case one of the parties in R_{sid} goes off-line, in which case potentially a future session will terminate before that party goes back online. In any case a party sends an invalid message, e.g., not well-formed, does not pass verification, or sends two messages for the same session and round, it is placed into M and ignored thereafter.

Crucially, parties $P_j \notin R_{\mathsf{sid}}$ who respond with $(\perp, r_j^{(\mathsf{sid}+1)})$ are also placed into $R_{\mathsf{sid}+1}$. This is important for guaranteed output delivery, allowing parties to go back online and catch-up, and allowing each party to participate in each round. However, along with sending this message, party j is required to send the responses $z_j^{(\mathsf{sid}')}$ for every sid' for which $j \in R_{\mathsf{sid}'}$ did not send its response. This avoids the issue of parties going offline and online back and forth and preventing any session from finalizing.

Finally, upon receiving a response $z_j^{(\mathsf{sid})}(R_{\mathsf{sid}})$ from each party in R_{sid}, each party can aggregate the signature shares and derive the signature $z = \sum_{j \in R_{\mathsf{sid}}} z_j^{(\mathsf{sid})}$.

By construction, every session will either succeed, add malicious actors to M, or will never terminate because a party in R_{sid} will be missing indefinitely. However, assuming the adversary controls $< N - t$ parties, and that every honest party eventually goes back online, the adversary can stall up to $n - t$ sessions by not responding. This is because whenever it does respond, we require it to also fill-in all previous sessions in which it was not available.

Therefore, in what remains, Protocol 6 is described under the assumption that the set S_1 is post-determined but S_2 is (effectively) predetermined, and equals S_1. The changes with respect to Protocol 4 are highlighted in light blue.

The proof for the following theorem is provided in Appendix E.5.

Theorem 7. *Protocol 5 UC-realizes the distributed key generation phase of $\mathcal{F}_{\mathtt{ThFHE}}$ (Functionality 1).*

C.3 DKG of Protocol 2

Finally, in order to comply with the optimized construction, Protocol 6 augments Protocol 5 by distributively generating the additional required evaluation keys. Specifically:

1. Each party P_i outputs a secret share $\mathsf{sk}_i^{\mathtt{TAHE}}$ of a secret decryption key $\mathsf{sk}^{\mathtt{TAHE}}$ with respect to some $\mathtt{TAHE}$ scheme E, and all parties output the corresponding public key $\mathsf{pk}^{\mathtt{TAHE}}$. This can be assumed to be generated by a DKG protocol of the underlying TAHE scheme E chosen, and we assume it to be available at setup for ease of exposition.
2. All parties output a key-switching key ks from the corresponding $\mathtt{RLWE}$ key to itself. This is already a by-product of generating the relinearization key and does not require further changes to the protocol.

PROTOCOL 5 (*DKG of Protocol 2 (Π_{DKG})*)

Setup: Each party $P_i \in \mathcal{P}$ retrieves $a, \{a_\ell\}_{\ell\in[0,\ell_0]} \leftarrow \mathcal{R}_Q$ from the Random Oracle $\mathcal{O}$. In addition, each party samples a public-private key pair and computes a corresponding ZKP for the underlying PVSS scheme.

1. **Round 1**: Each party $P_i \in \mathcal{P}$
   (a) Samples $s_i, e_i, \{e_{i,\ell}\}_{\ell\in[0,\ell_0]} \leftarrow \chi_{\mathsf{E}}$.
   (b) Calculates $b_i = \mathsf{E}_{s_i}(0, a, e_i)$ and $b_i^\ell = \mathsf{E}_{s_i}(w^{L-\ell}s_i, a_\ell, e_{i,\ell})$ for $\ell \in [0, \ell_0]$.
   (c) Generates a ZK proof $\Pi_i(b_i, \{b_i^\ell\}_{\ell\in[0,\ell_0]}; s_i, e_i, \{e_{i,\ell}\}_{\ell\in[0,\ell_0]})$ of correct encryption of s_i and its powers of w decomposition.
   (d) Calls protocol $\Pi_{\mathsf{PVSS}}.\mathsf{Dist}(s_i)$ to verifiably share its secret s_i.
   (e) Broadcasts $(b_i, \{b_i^\ell\}_{\ell\in[0,\ell_0]}, \Pi_i, \{v_{i,j}\}_{j\in[N]})$ to all parties.
   (f) Generates and broadcasts a proof $\Pi_i'(b_i, F_{i,0}; s_i)$ that binds b_i to the committed s_i of the PVSS.
2. **Round 2**: Upon receiving valid messages from a set S_1 of $t+1$ parties, each party $P_i \in S_1$:
   (a) Computes $b = \sum_{j\in S_1} b_j$, $b^\ell = \sum_{j\in S^*} b_j^\ell$ for $\ell \in [0, \ell_0]$, $\mathsf{sk}_i = \sum_{j\in S^*} s_{j,i}$ and $\mathsf{vk}_{j'} = \sum_{j\in S^*} v_{j,j'}$ for each $j' \in [N]$. Denote $\mathsf{pk} = (a, b)$ and $\mathsf{ks}_\ell' = (a_\ell, b^\ell)$.
   (b) Samples $\{u_{i,\ell}, e_{i,\ell}', e_{i,\ell}''\}_{\ell\in[0,\ell_0]} \leftarrow \chi_{\mathsf{E}}$.
   (c) Sets $a_{i,\ell}^0 = u_{i,\ell} \cdot a + e_{i,\ell}'$ and $b_{i,\ell}^0 = 0 + e_{i,\ell}'' + u_{i,\ell} \cdot b$.
   (d) Computes $(a_{\mathsf{lk},i}^\ell, b_{\mathsf{lk},i}^\ell) = \lambda_{i,0}^{S_1}\mathsf{sk}_i \cdot (a_\ell, b^\ell) + (a_{i,\ell}^0, b_{i,\ell}^0)$.
   (e) Generates a corresponding ZK proof $\Pi_i^{\mathsf{lk}}(\{(a_{\mathsf{lk},i}^\ell, b_{\mathsf{lk},i}^\ell)\}_{\ell\in[0,\ell_0]}, \mathsf{vk}_i; \mathsf{sk}_i, \{(u_{i,\ell}, e_{i,\ell}'', e_{i,\ell}')\}_{\ell\in[0,\ell_0]})$.
   (f) Broadcasts $(\{(a_{\mathsf{lk},i}^\ell, b_{\mathsf{lk},i}^\ell)\}_{\ell\in[0,\ell_0]}, \Pi_i^{\mathsf{lk}})$.

   Output: Upon receiving valid messages from each party $P_j \in S_1$, each party $P_i \in \mathcal{P}$
   (a) Calculates $\mathsf{lk}_\ell = \sum_{j\in S_1} (a_{\mathsf{lk},j}^\ell, b_{\mathsf{lk},j}^\ell)$ for each $\ell \in [0, \ell_0]$.
   (b) Outputs $(\mathsf{pk}, \mathsf{lk}, \{\mathsf{vk}_j\}_{j\in[N]}; \mathsf{sk}_i)$.

3. In addition, all parties output the corresponding encryption of the randomizers for ks, $\mathsf{ct}_u^{\mathsf{ks}} = \mathsf{E}(\mathsf{pk}^{\mathtt{TAHE}}, u_{\mathsf{ks}})$. This can be computed alongside the computation of ks, and the two are tied by a proper ZKP.

The proof for the following theorem is provided in Appendix E.6.

Theorem 8. *Protocol 6 UC-realizes the distributed key generation phase of $\mathcal{F}_{\mathtt{ThFHE}}$ (Functionality 1).*

D Discussion on Theorem 1 and Experimental Analysis

In Theorem 1, we establish a bound on the size of $\Delta\lambda_{(i_0,j_0),0}^S$. In this section, we provide further insights into this result and propose potential improvements based on heuristics supported by experimental results.

PROTOCOL 6 (*DKG of Protocol 2 (Π_{DKG})*)

Setup: Each party $P_i \in \mathcal{P}$ retrieves $a, \{a_\ell\}_{\ell\in[0,\ell_0]} \leftarrow \mathcal{R}_Q$ from the Random Oracle $\mathcal{O}$. In addition, each party samples a public-private key pair and computes a corresponding ZKP for the underlying PVSS scheme. In addition, each party participates in $\mathtt{TAHE}.\Pi_{\mathsf{DKG}}$, a DKG protocol for a $\mathtt{TAHE}$ scheme, and retrieves $\mathsf{sk}_i^{\mathtt{TAHE}}$, its private decryption share, and the corresponding public key $\mathsf{pk}^{\mathtt{TAHE}}$.

1. **Round 1**: Each party $P_i \in \mathcal{P}$
   (a) Samples $s_i, e_i, \{e_{i,\ell}\}_{\ell\in[0,\ell_0]} \leftarrow \chi_{\mathsf{E}}$.
   (b) Calculates $b_i = \mathsf{E}_{s_i}(0, a, e_i)$ and $b_i^\ell = \mathsf{E}_{s_i}(w^{L-\ell}s_i, a_\ell, e_{i,\ell})$ for $\ell \in [0, \ell_0]$.
   (c) Generates a ZK proof $\Pi_i(b_i, \{b_i^\ell\}_{\ell\in[0,\ell_0]}; s_i, e_i, \{e_{i,\ell}\}_{\ell\in[0,\ell_0]})$ of correct encryption of s_i and its powers of w decomposition.
   (d) Calls protocol $\Pi_{\mathsf{PVSS}}.\mathsf{Dist}(s_i)$ to verifiably share its secret s_i.
   (e) Broadcasts $(b_i, \{b_i^\ell\}_{\ell\in[0,\ell_0]}, \Pi_i, \{v_{i,j}\}_{j\in[N]})$ to all parties.
   (f) Generates and broadcasts a proof $\Pi_i'(b_i, F_{i,0}; s_i)$ that binds b_i to the committed s_i of the PVSS.
2. **Round 2**: Upon receiving valid messages from a set S_1 of $t+1$ parties, each party $P_i \in S_1$:
   (a) Computes $b = \sum_{j\in S_1} b_j$, $b^\ell = \sum_{j\in S^*} b_j^\ell$ for $\ell \in [0, \ell_0]$, $\mathsf{sk}_i = \sum_{j\in S^*} s_{j,i}$ and $\mathsf{vk}_{j'} = \sum_{j\in S^*} v_{j,j'}$ for each $j' \in [N]$. Denote $\mathsf{pk} = (a, b)$ and $\mathsf{ks}_\ell' = (a_\ell, b^\ell)$.
   (b) Samples $\{u_{i,\ell}, e_{i,\ell}', e_{i,\ell}''\}_{\ell\in[0,\ell_0]} \leftarrow \chi_{\mathsf{E}}$ and $\{\bar{u}_{i,\ell}, \bar{e}_{i,\ell}', \bar{e}_{i,\ell}''\}_{\ell\in[0,\ell_0]} \leftarrow \chi_{\mathsf{E}}$.
   (c) Sets $a_{i,\ell}^0 = u_{i,\ell} \cdot a + e_{i,\ell}'$ and $b_{i,\ell}^0 = 0 + e_{i,\ell}'' + u_{i,\ell} \cdot b$, and $\bar{a}_{i,\ell} = \bar{u}_{i,\ell} \cdot a + \bar{e}_{i,\ell}'$, $\bar{b}_{i,\ell} = \lambda_{i,0}^{S_1} w^{L-\ell} \mathsf{sk}_i \; \bar{e}_{i,\ell}'' + \bar{u}_{i,\ell} \cdot b$, and $\mathsf{ct}_{u,i}^{\mathsf{ks}} = \mathsf{E}(\mathsf{pk}^{\mathtt{TAHE}}, (\bar{u}_i, \ell)_\ell)$. It also sets sets $\mathsf{ks}_{i,\ell} = \{(\bar{a}_{i,\ell}, \bar{b}_{i,\ell})\}$ for each $\ell \in [0, \ell_0]$
   (d) Computes $(a_{\mathsf{lk},i}^\ell, b_{\mathsf{lk},i}^\ell) = \lambda_{i,0}^{S_1} \mathsf{sk}_i \cdot (a_\ell, b^\ell) + (a_{i,\ell}^0, b_{i,\ell}^0)$.
   (e) Generates a corresponding ZK proof $\Pi_i^{\mathsf{lk}}(\{(a_{\mathsf{lk},i}^\ell, b_{\mathsf{lk},i}^\ell)\}_{\ell\in[0,\ell_0]}, \mathsf{ks}_i, \mathsf{ct}_{u,i}^{\mathsf{ks}}, \mathsf{vk}_i; \mathsf{sk}_i, \{(u_{i,\ell}, e_{i,\ell}'', e_{i,\ell}')\}_{\ell\in[0,\ell_0]})$.
   (f) Broadcasts $(\{(a_{\mathsf{lk},i}^\ell, b_{\mathsf{lk},i}^\ell)\}_{\ell\in[0,\ell_0]}, \mathsf{ks}_i, \mathsf{ct}_{u,i}^{\mathsf{ks}}, \Pi_i^{\mathsf{lk}})$.

Output: Upon receiving valid messages from each party $P_j \in S_1$, each party $P_i \in \mathcal{P}$
   (a) Calculates $\mathsf{ks}_\ell = \sum_{j\in S_1} \mathsf{ks}_{i,\ell}$ and $\mathsf{ct}_u^{\mathsf{ks}} = \sum_{j\in S_1} \mathsf{ct}_{u,i}^{\mathsf{ks}}$.
   (b) Calculates $\mathsf{lk}_\ell = \sum_{j\in S_1} (a_{\mathsf{lk},j}^\ell, b_{\mathsf{lk},j}^\ell)$ for each $\ell \in [0, \ell_0]$.
   (c) Outputs $(\mathsf{pk}, \mathsf{lk}, \mathsf{ks}, \mathsf{ct}_u^{\mathsf{ks}}, \{\mathsf{vk}_j\}_{j\in[N]}; \mathsf{sk}_i)$.

We observe three points that Theorem 1 does not take into account, that can each lead to an overestimation of the parameters: (i) it does not take advantage of any ring structure, meaning it is a general result applicable to polynomials

with integer coefficients; (ii) the lemma analyzes the worst-case subset of parties $S \subset \mathcal{P}$ is chosen. These observations suggest that there is potential for improving the result; (iii) The proof relies on the sub-multiplicativity of the $\|\cdot\|_1$ norm, assuming equality holds in all cases.

The first question that arises is whether, given the proven upper bound and the worst-case scenario, we can also establish a lower bound for the worst-case scenario. Specifically, for $t = \frac{N}{2}$ and S which consists of x^j for $j = 0$ to $\frac{N}{2}$, the term $\Delta \cdot \lambda^S_{(0,0),0}$ leaves us with expression of the form $(\Pi_{j=1}^{N/2} x^j + 1) \cdot \Pi_{j=1}^{N/6}(x^{2j+1} - 1)$, which suggests a lower bound of the order of magnitude of $2^{N/2}$.

Regarding point (i), note that the infinity norm of a polynomial is bounded by its maximum absolute value on the complex unit circle. However, when considering the norm in the canonical embedding (refer to [LPR13] for details), we move from maximizing over the entire unit circle to a maximum over a discrete set of points on the unit circle, specifically the roots of the cyclotomic polynomial. A lower bound, possibly derived using results from Sudler's product (see the introduction of [ATZ23] for a survey on this topic), similar to [Wri64], could be used to refine the result. That being said, as n grows larger, due to continuity, the difference between the two approaches is likely not significant.

Another observation is that during threshold decryption, but before performing recombination, the parties could check that the size of the added error is not too large before proceeding with the computation. This would ensure that, even if there exist "bad" subsets that cannot be used for decryption, it would only lead to a delay or a denial of service. Indeed, no secret information is revealed upon failure. Formally, Theorem 3 holds even when the conditions for correctness of decryption in Theorem 2 do not hold.

Therefore, one could select a lower Q such that only a portion of the authorized subsets of parties are capable of decryption. Assuming the attacker cannot delay messages and the subset is chosen randomly, there is some success probability for decryption. If the first $t+1$ decryptors do not consist a proper subset, the parties can simply wait for more shares. Simulations we conducted, depicted in Figure 7 suggest that for a random subset of decryptors, the average size of the added error is approximately $\frac{3}{25} \cdot N$.

More importantly, the assumption that the subset S is random is not valid in the presence of malicious parties, who could influence the distribution of the subset. An adversary controlling $f > n - t$ of the parties may skew the distribution, making it less likely that a valid subset is found. However, we assume in this work that $f < n - t$ for secure broadcast channels and guaranteed output delivery, in which case such attacks are not feasible. In addition any such improvement may blunted by the factor $1.2^{\frac{2}{3}N}$ appearing in Theorem 2 and the factor 2^t appearing in Theorem 3.

E Omitted Proofs

E.1 Missing Upper Bounds Proofs

Lemma 3. $\|\Delta\|_\infty \leq 1.2^{\frac{2}{3}N}$

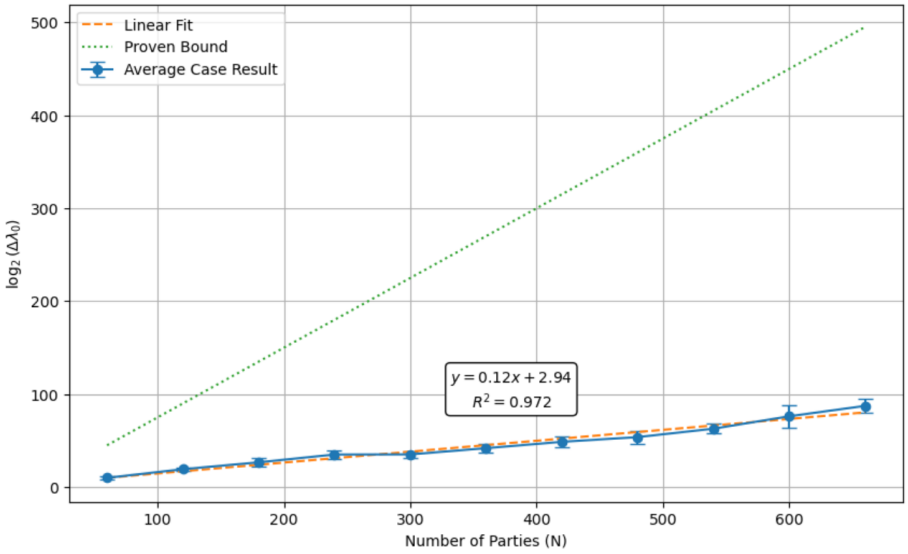


Fig. 7: Bit-length of the adjusted Lagrange coefficient norms $\left\|\Delta\lambda_{0,i}^{S}\right\|_{\infty,\Phi(n)}$ scale with the number of parties N. The blue trend averages over 10 samples of subsets of parties of size $|S| = t + 1$, where $t = \lfloor N/3 \rfloor$. Each sample considers the interpolation point of maximal norm. The green line depicts the proven upper bound.

Proof. According to [Wri64], we have $\left\|\prod_{e\in[N]} x^e - 1\right\|_\infty \leq 1.2^N$. Replacing x with x^2, we get that $\|\Pi_1\| = \left\|\prod_{e\in[N/2-1]} x^{2e} - 1\right\|_\infty \leq 1.2^{\frac{N}{2}-1}$ and similarly $\|\Pi_1\| = \left\|\prod_{e\in[N/6]} x^{2e} - 1\right\|_\infty \leq 1.2^{\frac{N}{6}}$. Therefore, as $\Delta = 2\Pi_1\Pi_2$, we conclude that $\|\Delta\|_\infty \leq 1.2^{\frac{2N}{3}}$. □

Corollary 1. *With the same notations as above, we have (1)* $\|\Delta\|_{\infty,\Phi_n} \leq \frac{N^2}{3n}1.2^{\frac{2}{3}N}$*; and (2)* $\left\|\Delta\cdot\lambda^S_{(i_0,j_0),0}\right\|_{\infty,\Phi_n} \leq \frac{N^2}{3n}2^{\frac{3}{4}N}$.

Proof. The degree of Δ is calculated as follows:

$$\sum_{i\in[\frac{N}{2}-1]} 2i + \sum_{i\in[\frac{N}{6}]} 2i = \frac{N}{2}(\frac{N}{2}-1) + (\frac{N}{6}+1)\frac{N}{6} = \frac{10}{36}N^2 - \frac{1}{3}N \leq \frac{10}{36}N^2$$

Therefore, by Lemma 1, $\|\Delta\|_{\infty,\Phi_n} \leq \frac{\deg(\Delta)}{n-1}\|\Delta\|_\infty \leq \frac{10N^2}{36(n-1)}\|\Delta\|_\infty \leq \frac{N^2}{3n}\|\Delta\|_\infty$. The last inequality requires $n > 2$, which holds since $n > N \geq 6$.

Next, the degree of $\Delta\cdot\lambda^S_{(i,j),0}$ equals the degree of the numerator minus the degree of the denominator. The degree of the numerator is the degree of Δ plus $\sum_{(i',j')\in S} j'$. The degree of the denominator is $\sum_{(i',j')\in S}\max(\{j,j'\})$. Therefore, the degree is given by:

$$\deg(\Delta) - \sum_{(i',j')\in S: j>j'} j - j' \leq \deg(\Delta)$$

Similarly, applying Lemma 1 we get that $\left\|\Delta\cdot\lambda^S_{(i,j),0}\right\|_{\infty,\Phi_n} \leq \frac{N^2}{3n}\left\|\Delta\cdot\lambda^S_{(i,j),0}\right\|_\infty$. □

Lemma 4. *For any* $U \in \binom{[N]}{t}$ *we have that* $\left\|\lambda^{U\cup\{0\}}_{0,(i_0,j_0)}\right\|_{1,\Phi} \leq 2^t$.

Proof. We have

$$\left\|\lambda^{U\cup\{0\}}_{0,(i,j)}\right\|_{1,\Phi} = \left\|\prod_{(i',j')\in U}\frac{(-1)^i x^j - (-1)^{i'}x^{j'}}{(-1)^{i'+1}x^{j'}}\right\|_{1,\Phi} = \left\|\prod_{(i',j')\in U}(-1)^i x^j - (-1)^{i'}x^{j'}\right\|_{1,\Phi}$$
$$\leq \prod_{(i',j')\in U}\left\|((-1)^i x^j - (-1)^{i'}x^{j'})\right\|_1 \leq 2^t,$$

Where the second equality is due to $x^{-j'} \equiv -x^{n-j'} \mod \Phi$, and the first inequality is due to the sub-multiplicativity the norm. □

E.2 Security of RLWE with Noise from $\Delta\cdot\chi_{\mathsf{E}}$

In this section we formally prove Remark 2. We refer to Definition 7 for the formal definition of the decisional RLWE assumption.

Proof. Given a distinguisher $\mathcal{D}_\alpha$ between $\mathcal{U}(\mathcal{R}_q^2)$ and $A_{\mathsf{s},\alpha}$ where the noise e is sampled from $\alpha\chi_{\mathsf{E}}$, we construct a distinguisher $\mathcal{D}_1$ between $\mathcal{U}(\mathcal{R}_q^2)$ and $A_{\mathsf{s},1}$ where the noise e is sampled from χ_{E}. Specifically, $\mathcal{D}_1$ invokes $\mathcal{D}_\alpha$ internally, and simulates its sample requests. Specifically, whenever $\mathcal{D}_1$ queries a sample, $\mathcal{D}_\alpha$ transfers it and retrieve a sample $(a, b) \in \mathcal{R}_q$, that depending on the challnger's secret bit b, is either drawn from $\mathcal{U}(\mathcal{R}_q^2)$ or from $A_{\mathsf{s},1}$. Then, $\mathcal{D}_1$ returns to $\mathcal{D}_\alpha$ the simulated sample $(a \cdot \alpha, b \cdot \alpha)$. At the end, $\mathcal{D}_1$ outputs whatever $\mathcal{D}_\alpha$ outputs.

Now, since $\langle \alpha, q \rangle = \mathcal{R}$, α is invertible in $\mathcal{R}_q$. Therefore, $\alpha\mathcal{U}(\mathcal{R}_q) \equiv \mathcal{U}(\mathcal{R}_q)$, which means $\alpha\mathcal{U}(\mathcal{R}_q^2) \equiv \mathcal{U}(\mathcal{R}_q^2)$, and $\alpha A_{\mathsf{s},1} = A_{\mathsf{s},\alpha}$. Namely, our simulator correctly transfroms samples from the RLWE challenger with parameters $(n, q, \chi_{\mathsf{s}}, \chi_{\mathsf{E}})$ to samples from the RLWE challenger with parameters $(n, q, \chi_{\mathsf{s}}, \alpha\chi_{\mathsf{E}})$, and therefore the distinguisher $\mathcal{D}_1$ achieves the exact same advantage as $\mathcal{D}_\alpha$.

E.3 Security Proof of Π_{PVSS} (Protocol 3)

Theorem 9. *Protocol 3 is fair and t-zero-knowledge if* $\sqrt{Q} > \frac{N^2}{3n} 2^{\frac{3}{4}N} \times t \times r_{\mathsf{E}} \times 2^{1+\sigma/2n}$.

Proof. **Fairness**: Assume by contradiction that there exists $S, S' \in \binom{[N]}{t+1}$ such that $s := \sum_{i \in S} \lambda_{0,i}^S s_i \neq \sum_{i' \in S} \lambda_{0,i'}^{S'} s_{i'} := s'$. By the verification of the secret shares in $\Pi_{\mathsf{PVSS}}.\mathsf{VerifyDist}()$, we know that $G_i := \sum_{k=0}^t F_k \alpha_i^k = \mathbf{com}(s_i; e_0^i, e_1^i)$ holds for each $i \in [N]$, where e_0^i, e_1^i are bounded. Applying Lagrange interpolation using S and S' respectively suggests that $\mathbf{com}(s; e_0^S, e_1^S) = \sum_{i \in S} \lambda_{0,i}^S G_i = \sum_{i' \in S'} \lambda_{0,i'}^{S'} G_{i'} = \mathbf{com}(s'; e_0^{S'}, e_1^{S'})$. Multiplying by Δ, we obtain $\mathbf{com}(\Delta s; \Delta e_0^S; \Delta e_1^S) = \mathbf{com}(\Delta s'; \Delta e_0^{S'}; \Delta e_1^{S'})$. By Corollary 1, the adjusted Lagrange coefficients have bounded norm $\left\|\Delta \cdot \lambda_{0,i}^S\right\|_{\infty, \Phi_n} \leq \frac{N^2}{3n} 2^{\frac{3}{4}N}$, and $\Delta e_0^S, \Delta e_1^S, \Delta e_0^{S'}, \Delta e_1^{S'}$ are the sums of t fresh noises multiplied by such adjusted Lagrange coefficients. Therefore $\left\|\Delta e_0^S\right\|_{\infty, \Phi(n)} \leq \frac{N^2}{3n} 2^{\frac{3}{4}N} \times t \times r_{\mathsf{E}}$, and similarly for $\Delta e_1^S, \Delta e_0^{S'}, \Delta e_1^{S'}$. Imposing $\frac{N^2}{3n} 2^{\frac{3}{4}N} \times t \times r_{\mathsf{E}} < \sqrt{Q} 2^{-1-\sigma/2n}$, we get a contradiction to the binding property of $\mathsf{COM}_{\mathsf{async}}$.

Zero-knowledge: Note that the distribution over $s \in \mathcal{R}_Q$ given a subset of t shares is uniform. It remains to simulate the public output of the dealer. However, the latter consists of computationally hiding commitments and a zk proof.

IND1-secrecy (Definition 6): Assume by contradiction that the adversary is able to gain an advantage in the game then it can either break the RLWE assumption (distinguish·ing the public keys, the commitments or the encryption from uniform) or distinguish between the simulated and real execution of the zk protocols both of which we assume are impossible.

E.4 Correctness Proof of Theorem 6

We begin by analyzing the correctness of Protocol 4.

Proof. The secret decryption key is defined as $\mathsf{sk} := \sum_{j \in S_1} s_j$, where $S_1 \subset \mathcal{P}$ consists of the first $t+1$ parties who sent a valid message in round 1. Note that this is well-defined, since the protocol is implemented on top of a consensus channel, providing agreement on the ordering of messages. Furthermore, denote $e := \sum_{j \in S_1} e_j$. Then:

$$b := \sum_{j \in S_1} b_j = \sum_{j \in S_1} (a s_j + \Delta e_j) = a \sum_{j \in S_1} s_j + \Delta \sum_{j \in S_1} e_j = a\mathsf{sk} + \Delta e$$

This demonstrates the correctness of the public key $\mathsf{pk} = (a, b)$.

Now, we prove that the relinearization key adheres to the expected format. First, denote $e^\ell := \sum_{j \in S_1} e_{j,\ell}$ for every $\ell \in [0, \ell_0]$. Then:

$$\begin{aligned} b^\ell := \sum_{j \in S_1} b_j^\ell &= \sum_{j \in S_1} (w^{L-\ell} s_j + a_\ell s_j + \Delta e_{j,\ell}) = w^{L-\ell}\mathsf{sk} + a_\ell \mathsf{sk} + \Delta e_\ell \\ &= \mathsf{E}_{\mathsf{sk}}(w^{L-\ell}\mathsf{sk}, a_\ell, \Delta e_\ell) \end{aligned}$$

Therefore, $\{(a_\ell, b^\ell)\}_\ell$ is an encryption of the decomposition of sk under sk, and can be thought of as a key-switching key from sk to itself.

Finally, we compute the ℓ-entry of the relinearization key. We denote by $u_\ell := \sum_{j \in S_2} (\Delta \lambda_{j,0}^{S_2}) u_{j,\ell}, e'_\ell := \sum_{j \in S_2} (\Delta \lambda_{j,0}^{S_2}) e'_{j,\ell}, e''_\ell := \sum_{j \in S_2} (\Delta \lambda_{j,0}^{S_2}) e''_{j,\ell}$. Note that due to Corollary 1, the adjusted Lagrange coefficients have bounded norms and therefore $\{u_\ell, e'_\ell, e''_\ell\}_\ell$ are all bounded. Additionally, we denote $a_\ell^0 := \sum_{j \in S_2} \lambda_{j,0}^{S_2} a_{j,\ell}^0, b_\ell^0 := \sum_{j \in S_2} \lambda_{j,0}^{S_2} b_{j,\ell}^0$. Therefore, we get:

$$\begin{aligned} a_{\mathsf{lk}}^\ell :&= \sum_{j \in S_2} \lambda_{j,0}^{S_2} a_{\mathsf{lk},j}^\ell = \sum_{j \in S_2} \lambda_{j,0}^{S_2} (\mathsf{sk}_j a_\ell + \Delta u_{j,\ell} a + \Delta e'_{j,\ell}) \\ &= a_\ell \sum_{j \in S_2} \lambda_{j,0}^{S_2} \mathsf{sk}_j + \sum_{j \in S_2} (\Delta \lambda_{j,0}^{S_2}) e'_{j,\ell} + a \sum_{j \in S_2} (\Delta \lambda_{j,0}^{S_2}) u_{j,\ell} \\ &= a_\ell \mathsf{sk} + e'_\ell + u_\ell a \end{aligned}$$

$$\begin{aligned} b_{\mathsf{lk}}^\ell :&= \sum_{j \in S_2} \lambda_{j,0}^{S_2} b_{\mathsf{lk},j}^\ell = \sum_{j \in S_2} \lambda_{j,0}^{S_2} (\mathsf{sk}_j b^\ell + \Delta u_{j,\ell} b + \Delta e''_{j,\ell}) \\ &= b^\ell \sum_{j \in S_2} \lambda_{j,0}^{S_2} \mathsf{sk}_j + \sum_{j \in S_2} (\Delta \lambda_{j,0}^{S_2}) e''_{j,\ell} + b \sum_{j \in S_2} (\Delta \lambda_{j,0}^{S_2}) u_{j,\ell} \\ &= b^\ell \mathsf{sk} + e''_\ell + u_\ell b \end{aligned}$$

Therefore,

$$\begin{aligned} b_{\mathsf{lk}}^\ell &= w^{L-\ell}\mathsf{sk}^2 + (a_\ell \mathsf{sk} + \Delta e_\ell)\mathsf{sk} + e''_\ell + u_\ell(a\mathsf{sk} + \Delta e) = \\ &= w^{L-\ell}\mathsf{sk}^2 + a_{\mathsf{lk}}^\ell \mathsf{sk} + [\Delta e_\ell \mathsf{sk} + e''_\ell + u_\ell \Delta e - e'_\ell \mathsf{sk}] = \mathsf{E}_{\mathsf{sk}}(w^{L-\ell}\mathsf{sk}^2, a_{\mathsf{lk}}^\ell, e_\ell^{\mathsf{lk}}), \end{aligned}$$

where $e_\ell^{\mathsf{lk}} = \Delta e_\ell \mathsf{sk} + e''_\ell + u_\ell \Delta e - e'_\ell \mathsf{sk}$ is bounded. This structure precisely aligns with the required format for the relinearization key.

E.5 Security Proof of Theorem 7

Proof. For the security proof, we observe all broadcast messages within our DKG construction are either in encrypted form or are transmitted via fundamental building blocks such as VSS and ZK proofs. These messages, based on the security property of these building blocks, do not reveal any information about the underlying secret. Additionally, we guarantee that any subset of t parties cannot reconstruct the original secret due to the application of the Shamir secret sharing scheme. Consequently, without collusion of $t+1$ parties, neither the secret key nor any inputs of other parties can be revealed. Moreover, potential malicious adversaries can be thwarted through the utilization of VSS and ZK Proofs.

Formally, we prove that Protocol 5 UC realizes the DKG phase of Functionality 1. We construct our UC simulation by applying a sequence of three hybrids T_0, T_1, T_2, starting from T_0, the view of the adversary in the real execution of the protocol, and ending up with T_1, the UC simulation.

- *Hybrid* 1*:* Let T_1 be the same as T_0, except for the broadcast message in Round 1. Upon receiving $\mathsf{pk} = (a, b), \mathsf{ks} = \{(a_\ell^{\mathsf{ks}}, b_\ell^{\mathsf{ks}})\}_\ell, \mathsf{lk} = \{(a_{\mathsf{lk}}^\ell, b_{\mathsf{lk}}^\ell)\}$ from $\mathcal{F}_{\texttt{ThFHE}}$, the simulator sets for each honest party $P_i \in \mathcal{P}$, $b_i = b + \mathsf{E}_{s_i}(0, a, \Delta e_i)$ and $b_{i,\ell}^{\mathsf{ks}} = b_\ell^{\mathsf{ks}} + \mathsf{E}_{s_i}(w^{L-\ell} s_i, a_\ell, \Delta e_\ell)$ for each $\ell \in [0, \ell_0]$. It then simulates the corresponding ZKPs Π_i, Π_i'.
  Upon receiving from the adversary $\mathcal{A}$ the subset S_1 of parties that participate in round 1, and the messages from each party in S_1 controlled by $\mathcal{A}$, the simulator proceeds as follows. First, it sets $\delta = |S_1|$, and extracts (s_j, e_j) from the ZKP of b_j of each malicious party.[8] It then sets $\varepsilon = \sum_{j \in S_1} (s_j, e_j)$ and sends $(\mathbf{bias}, \delta, \varepsilon)$ to $\mathcal{F}_{\texttt{ThFHE}}$.
- *Hybrid* 2*:* Let T_2 be the same as T_1, except the broadcast message in Round 2. The simulator retrieves lk from the ideal functionality. It then computes $\mathsf{lk}_{\mathcal{A}} := \sum_{j in U_1} \lambda_{j,0}^{S_1} \mathsf{sk}_i \cdot (a_\ell, b^\ell)$, the part of the adversary of the relinearization key, where $U_1 \subseteq S_1$ is the subset of parties controlled by the $\mathcal{A}$. It then samples $(a_{\mathsf{lk},i}^\ell, b_{\mathsf{lk},i}^\ell) \leftarrow \mathcal{R}_Q$ uniformly at random for each honest party in S_1, and broadcasts it. For one honest party $i^* \in S_1$, it broadcasts instead $\mathsf{lk} - \mathsf{lk}_c A - \sum_{j \in S_1 \setminus U_1} (a_{\mathsf{lk},i}^\ell, b_{\mathsf{lk},i}^\ell)$. The simulator simulates the corresponding ZKPs Π_i^{lk} of each honest party.

First, we claim that the transcript of the first round for hybrids T_0, T_1 is indistinguishable. Indeed, the PVSS execution is identical, the ZKPs are by definition zero-knowledge and therefore their simulation is indistinguishable from an honestly generated proof. Lastly, the ciphertexts are indistinguishable since E is CPA-secure. Then, the transcript of the first round for hybrids T_1, T_2 is identical, and so it remains to show that their second round is indistinguishable. Again, the ZKPs are indistinguishable by definition. Also, by CPA security, the ciphertexts sent are indistinguishable. Finally, due to the soundness of the ZKPs

[8] We only require the ZKP of b_j to be UC-extractable. This does not significantly affect performance, since the proofs for the relinearization keys need not be UC-extractable.

that are received from $\mathcal{A}$, we know that the output lk' equals to lk plus an encryption of 0 with a bounded noise. This concludes the proof.

E.6 Security Proof of Theorem 8

The UC-simulation of this protocol is very similar to the one described above. We will only discuss the difference below.

Essentially, the simulation of Protocol 6 must also simulate the parts colored in yellow, responsible for generating $\mathsf{ks}, \mathsf{ct}_u^{\mathsf{ks}}$. This is done as follows. We assume evk received from $\mathcal{F}_{\mathtt{ThFHE}}^{\mathsf{preprocess}}$ contains another key-switching key ks and a corresponding. Similarly to the simulation of lk, the simulator retrieves $\mathsf{ks}, \mathsf{ct}_u^{\mathsf{ks}}$ from the ideal functionality, and samples random values for the honest parties, that sum up to $\mathsf{ks} - \mathsf{ks}_{\mathcal{A}}$ and $\bar{ct}_u^{\mathsf{ks}} - \mathsf{ct}_{u,\mathcal{A}}^{\mathsf{ks}}$. This cancels-out the contribution of the adversary, up to encryptions of zeros with bounded noise. Thus the final $\mathsf{ct}_u^{\mathsf{ks}}$ is an encryption of a random value known to the simulator.

Finally, it remains to show that adding $\mathsf{ct}_u^{\mathsf{ks}}$ to evk is secure. This follows from the CPA-security of $\mathtt{TAHE}.\mathsf{E}$, suggesting it is indistinguishable from an encryption of zeros.

E.7 Security Proof of Theorem 3

Proof. Let ct be a well formed ciphertext i.e. $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ where $\mathsf{ct}_2 = \mathsf{pt} + \mathsf{ct}_1 \cdot \mathsf{sk} + \Delta \cdot e$ and $\|e\|_{\infty,\Phi} \leq B$. If $\frac{Q}{P} > nN\lceil\frac{N^2}{n}\rceil(r_{\mathsf{D}} 2^{\frac{3}{4}N} + B \cdot 1.2^{\frac{2}{3}N}) > nN[N^2/n]2^{\frac{3}{4}N}$:

$$\begin{aligned}
\mathsf{pt} = \mathsf{ct}_2 - \sum_{(i',j') \in S_{i,j}} \lambda_{(i',j'),0}^{S_{i,j}} \mathsf{ds}_{i',j'} &= \mathsf{ct}_2 - \sum_{(i',j') \in S_{i,j}} \lambda_{(i',j'),0}^{S_{i,j}} (\mathsf{ct}_1 \cdot \mathsf{sk}_{i',j'} + p\Delta e_{i',j'})) \\
&= \mathsf{ct}_2 - \mathsf{ct}_1 \mathsf{sk} + p \overbrace{\sum_{(i',j') \in S_{i,j}} \lambda_{(i',j'),0}^{S_{i,j}} \Delta e_{i',j'}}^{\bar{e}} \\
&= \mathsf{pt} + p(\Delta e) + p\bar{e} = \mathsf{pt} + p(\overbrace{\Delta e + \bar{e}}^{\tilde{e}})
\end{aligned}$$

Our goal now is to prove that $\tilde{e}$ is small enough to enable decryption. According to Corollary 1, we have $\left\|\Delta\lambda_{(i',j'),0}^{S}\right\|_{\infty,\Phi_n} \leq \frac{N^2}{3n} 2^{\frac{3}{4}N}$ which, by regular norm bounds, implies $\left\|\Delta\lambda_{(i',j'),0}^{S} e_{i',j'}\right\|_{\infty,\Phi_n} \leq r_{\mathsf{D}} n \frac{N^2}{3n} 2^{\frac{3}{4}N}$. Utilizing the triangle inequality, this yields $\|\tilde{e}\|_{\infty,\Phi_n} \leq r_{\mathsf{D}} nN \frac{N^2}{3n} 2^{\frac{3}{4}N}$. Similarly, we find that $\|\Delta e\|_{\infty,\Phi_n} \leq Bn\frac{N^2}{3n} 1.2^{\frac{2}{3}N}$. Consequently, in every coefficient, $\mathsf{pt} + p\tilde{e}$ does not modulate by Q, and thus, taking $\bmod p$ results in pt as needed.

Next, we demonstrate the security of threshold decryption. Consider a subset $U \subset \mathcal{P}$ where $|U| = t$. We proceed to outline a Simulator $\mathcal{S}$:

1. $\mathcal{S}$ simulates the DKG phase by calling the UC simulator described in Section C. As a result, $\mathcal{S}$ gets:

(a) The public key pk.

(b) The secret decryption shares of the adversary $\{\mathsf{sk}_{i,j}\}_{(i,j)\in U}$.

2. $\mathcal{S}$ then simulates threshold decryption as follows:

(a) Upon receiving a ciphertext ct to decrypt $\mathcal{S}$ calculates decryption shares for the corrupted parties $\mathsf{ds}_{i,j} = \mathsf{sk}_{i,j} \cdot \mathsf{ct}_1 + e_{i,j}$ where $e_{i,j} \leftarrow \chi_{\mathsf{D}}$.

(b) $\mathcal{S}$ sends $(\textbf{decrypt}, \mathsf{ssid}, \mathsf{ct}, P_{i,j})$ to $\mathcal{F}_{\texttt{ThFHE}}$ for each party. It then receives pt.

(c) for every $(i,j) \notin U$ it calculates

$$\mathsf{ds}_{i,j} = \lambda^{U\cup\{0\}}_{(i,j),0}(\mathsf{ct}_2 - \mathsf{pt}) + (\sum_{(i',j')\in U} \lambda^{U\cup\{0\}}_{(i',j'),(i,j)}\mathsf{sk}_{i',j'})\mathsf{ct}_1 + \Delta e_{i,j}$$

It uses the simulator of Π^{ds} to create "fake" proofs $\Pi^{\mathsf{ds}}_{i,j}$

(d) $\mathcal{S}$ sends $(\mathsf{ds}_{i,j}, \Pi^{\mathsf{ds}}_{i,j})$ to $\mathcal{A}$.

(e) Upon receiving $\mathsf{ds}_{i,j}, \Pi^{\mathsf{ds}}_{i,j}$ from a corrupted party the simulator verifies the correctness of the decryption share (this can be done by verifying the proof or by subtracting $c_1 \cdot \mathsf{sk}_{i,j}$ and check that only a small error remains.) If the verification is successful it send **continue** to the ideal functionality. Else it sends $(\textbf{abort}, P_{i,j})$ for a party that failed the check. It also broadcasts to every party in U $(\textbf{cheater}, P_{i,j})$.

We want to prove that the ideal and real executions are indistinguishable. The messages in the real execution are $(\mathsf{pk}, \{\mathsf{sk}_{i,j}\}_{(i,j)\in U}, \{(\mathsf{ds}_{i,j}, \Pi^{\mathsf{ds}}_{i,j})\}_{(i,j)\notin U})$. The first two messages are indistinguishable since we rely on the security of the $\mathcal{F}_{DKG}$. Note that $\mathsf{ct}_2 = \mathsf{pt} + \mathsf{sk} \cdot \mathsf{ct}_1 + e$, thus we have

$$\begin{aligned}
\mathsf{ds}_{i,j} &= \lambda^{U\cup\{0\}}_{0,(i,j)}(\mathsf{sk} \cdot \mathsf{ct}_1 + e) + (\sum_{(i',j')\in U} \lambda^{U\cup\{0\}}_{(i',j'),(i,j)}\mathsf{sk}_{i',j'})\mathsf{ct}_1 + \Delta e_{i,j} \\
&= (\sum_{(i',j')\in U\cup\{0\}} \lambda^{U\cup\{0\}}_{(i',j'),(i,j)}\mathsf{sk}_{i',j'})\mathsf{ct}_1 + \Delta e_{i,j} + \Delta\lambda^{U\cup\{0\}}_{0,(i,j)} e \\
&= \mathsf{sk}_{i,j}\mathsf{ct}_1 + \Delta(e_{i,j} + \lambda^{U\cup\{0\}}_{0,(i,j)} e)
\end{aligned}$$

By Lemma 4 we have that $\left\|\lambda^{U\cup\{0\}}_{0,(i,j)}\right\|_1 \leq 2^t$. If $r_{\mathsf{D}} > 2^t 2^\sigma n r_{\mathsf{E}}$, the statistical distance between $e_{i,j} + \lambda^{U\cup\{0\}}_{0,(i,j)} e$ and $e_{i,j}$ is at least 2^σ which gives statistical indistinguishability as needed. We want to show that the output is the same i.e. that we have correct decryption in the ideal word. The adversary U may choose any subset of the parties S to preform decryption, which is

$$
\begin{aligned}
&\sum_{(i',j')\in S} \lambda^{S}_{(i',j'),0}\mathsf{ds}_{i',j'} \\
&= \sum_{(i',j')\in S\cap U} \lambda^{S}_{(i',j'),0}\mathsf{ds}_{i',j'} + \sum_{(i',j')\in S\cap U^c} \lambda^{S}_{(i',j'),0}\mathsf{ds}_{i',j'} \\
&= \sum_{(i',j')\in S\cap U} \lambda^{S}_{(i',j'),0}(\mathsf{sk}_{i',j'}\mathsf{ct}_1 + \Delta e_{i',j'}) + \sum_{(i',j')\in S\cap U^c} \lambda^{S}_{(i',j'),0}(\mathsf{sk}_{i',j'}\mathsf{ct}_1 + \Delta e_{i',j'} + \Delta\lambda^{U\cup\{0\}}_{0,(i',j')}e) \\
&= \mathsf{sk}\cdot\mathsf{ct}_1 + \sum_{(i',j')\in S} \Delta\lambda^{S}_{(i',j'),0}e_{i',j'} + \sum_{(i',j')\in S\cap U^c} \Delta\lambda^{S}_{(i',j'),0}\lambda^{U\cup\{0\}}_{0,(i',j')}e
\end{aligned}
$$

Taking $\mathsf{ct}_2 - \sum_{(i',j')\in S} \lambda^{S}_{i',j',0}\mathsf{ds}_{i',j'}$ gives

$$
\mathsf{pt} + \sum_{(i',j')\in S} \Delta\lambda^{S}_{(i',j'),0}e_{i',j'} + \sum_{(i',j')\in S\cap U^c} \Delta\lambda^{S}_{(i',j'),0}\lambda^{U\cup\{0\}}_{0,(i',j')}e + \Delta e \tag{1}
$$

Bounding each term similarly to the proof of correctness gives

$$
\left\| \sum_{(i',j')\in S} \Delta\lambda^{S}_{(i',j'),0}e_{i',j'} \right\|_{\infty,\Phi} \leq r_{\mathsf{D}} n N \frac{N^2}{3n} 2^{\frac{3}{4}N}
$$

We also have

$$
\left\| \sum_{(i',j')\in S\cap U^c} \Delta\lambda^{S}_{(i',j'),0}\lambda^{U\cup\{0\}}_{0,(i',j')}e \right\|_{\infty,\Phi} \leq r_{\mathsf{E}} n N \lceil\frac{N^2}{n}\rceil 2^{\frac{3}{4}N} 2^{t}
$$

The last term Δe in Equation (1) is negligible. Note that the second term and the third term are negligible in the first term, which is the one present in the real world as well. Therefore, we can conclude that except with negligible probability, if correct decryption occurs in the real world, it also occurs in the simulation.

E.8 Security Proof of Theorem 5

Before starting the proof we present the threshold decryption with preprocess functionality $\mathcal{F}^{\mathsf{preprocess}}_{\mathtt{ThFHE}}$ (Functionality 2).

We also present functionality for a TAHE scheme $\mathcal{F}_{\mathsf{TAHE}}$ (Functionality 3) which will be used for modeling the TAHE in the protocol.

We are now ready to move the proof:

Proof. The correctness of Protocol 2 is straightforward given Theorem 2 and the correctness of the TAHE scheme alongside with the observation that the key switching (Step 2, i) and re-randomization (Step 2, ii) do not change the underlying plaintext.

To prove the security of our threshold decryption, we consider a subset $U \subset \mathcal{P}$ where $|U| = t$. We proceed to outline a Simulator $\mathcal{S}$:

FUNCTIONALITY 2. (ThFHE- $\mathcal{F}_{\mathit{ThFHE}}^{\mathit{preprocess}}$)

PARAMETERS: a set of N parties $\mathcal{P} = \{P_i\}_{i\in[N]}$, a threshold $t \leq N/3$, an adversary $\mathcal{A}$ controlling a subset of the parties $\mathcal{P}_\mathcal{A} \subset \mathcal{P}$ ($|\mathcal{P}_\mathcal{A}| \leq t$), an encryption scheme $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ with an *affine key-homomorphism.*
BEHAVIOR:

1. **Setup**: Upon receiving a command $(\mathbf{keygen}, \mathsf{sid}, P_i)$ from party $P_i \in \mathcal{P}$, send to $\mathcal{A}$ and record $(\mathbf{keygen}, \mathsf{sid}, P_i)$. After recording $t+1$ requests for a given sid, generate a key triplet $(\mathsf{pk}, \mathsf{evk}; \mathsf{sk}) \leftarrow \mathsf{G}(1^\kappa)$, and send $\mathsf{pk}, \mathsf{evk}$ to $\mathcal{A}$.
   - Upon receiving $(\mathbf{bias}, \mathsf{sid}, \delta, \varepsilon)$ from $\mathcal{A}$, set $\mathsf{sk} \leftarrow \delta \cdot \mathsf{sk} + \varepsilon$, update the public key pk and evk accordingly.
   - Record $(\mathsf{sid}, \mathsf{pk}, \mathsf{evk}; \mathsf{sk})$ and send $(\mathsf{pk}, \mathsf{evk}, \|\delta\|_\infty, \|\varepsilon\|_\infty)$ to all parties.
2. **preprocess** Upon receiving a command $(\mathsf{preprocess}, \mathsf{sid}, \mathsf{ssid}, P_i)$ for $\mathsf{ssid} = (\mathsf{sid}, \ldots)$, and P_i that is not recorded, if there is a record of $(\mathsf{sid}; \mathsf{sk})$, send to $\mathcal{A}$ and record $(\mathsf{ssid}, \mathsf{ct}, P_i)$. Upon recording $t+1$ requests, compute an encryption of zero $\mathsf{ct}_\mathcal{F}^\mathsf{ssid}$ with randomizer $u_\mathcal{F}^\mathsf{ssid} \leftarrow \hat{\chi}_\mathsf{E}$, record $(\mathsf{sid}, \mathsf{ssid}, \mathsf{ct}_\mathcal{F}^\mathsf{ssid}, u_\mathcal{F}^\mathsf{ssid})$ and send $\mathsf{ct}_\mathcal{F}^\mathsf{ssid}$ to $\mathcal{A}$.
3. **Decryption**: Upon receiving a command $(\mathbf{decrypt}, \mathsf{ssid}, \mathsf{ct}', P_i)$, for $\mathsf{ssid} = (\mathsf{sid}, \ldots)$, and P_i that is not recorded, if there is a record of $(\mathsf{sid}; \mathsf{sk})$, send to $\mathcal{A}$ and record $(\mathsf{ssid}, \mathsf{ct}', P_i)$. Upon recording $t+1$ requests, send $(\mathbf{decrypt}, \mathsf{sid}, \mathsf{ssid})$ to $\mathcal{A}$ which responds with $(\mathbf{decrypt}, \mathsf{sid}, \mathsf{ssid}, N_\mathsf{shift})$ for $1 \leq N_\mathsf{shift} \leq t$ compute the the correspondent key switched ciphertext $\mathsf{ct} = (\mathsf{ks}_0\mathsf{ct}'_1 + \mathsf{ct}'_0, \mathsf{ks}_1\mathsf{ct}'_1) + N_\mathsf{shift} \cdot \mathsf{ct}_\mathcal{F}^\mathsf{ssid}$ and output $u_\mathsf{ks}^\mathsf{ssid} + N_\mathsf{shift} \cdot u_\mathcal{F}^\mathsf{ssid}$ where $u_\mathsf{ks}^\mathsf{ssid} = \mathsf{ct}'_1 \odot u_\mathsf{ks}$.

FUNCTIONALITY 3. (TAHE- $\mathcal{F}_{\mathit{TAHE}}$)

PARAMETERS: a set of N parties $\mathcal{P} = \{P_i\}_{i\in[N]}$, a threshold t, an adversary $\mathcal{A}$ controlling a subset of the parties $\mathcal{P}_\mathcal{A} \subset \mathcal{P}$ ($|\mathcal{P}_\mathcal{A}| \leq t$), a AHE scheme $(\mathsf{G}, \mathsf{E}, \mathsf{D}, \oplus)$.
BEHAVIOR:

1. **Setup**: Upon receiving a command $(\mathbf{keygen}, \mathsf{sid}, P_i)$ from party $P_i \in \mathcal{P}$, send to $\mathcal{A}$ and record $(\mathbf{keygen}, \mathsf{sid}, P_i)$. After recording $t+1$ requests for a given sid, generate a key tuple $(\mathsf{pk}; \mathsf{sk}) \leftarrow \mathsf{G}(1^\kappa)$, and send pk to $\mathcal{A}$.
   - Upon receiving $(\mathbf{continue}, \mathsf{sid}, \mathcal{P}'_\mathcal{A})$ from $\mathcal{A}$, if $\mathcal{P}'_\mathcal{A} \cup \mathcal{P}_\mathcal{A} = \emptyset$ record $(\mathsf{sid}, \mathsf{pk}; \mathsf{sk})$ and send $(\mathsf{sid}, \mathsf{pk})$ to all parties. Otherwise send $(\mathsf{sid}, \mathbf{abort}, \mathcal{P}'_\mathcal{A} \cup \mathcal{P}_\mathcal{A})$ to all parties and restart the execution.
2. **Decryption**: Upon receiving a command $(\mathbf{decrypt}, \mathsf{ssid}, \mathsf{ct}, P_i)$, for $\mathsf{ssid} = (\mathsf{sid}, \ldots)$, and P_i that is not recorded, if there is a record of $(\mathsf{sid}; \mathsf{sk})$, send to $\mathcal{A}$ and record $(\mathsf{ssid}, \mathsf{ct}, P_i)$. Upon recording $t+1$ requests, compute the plaintext $\mathsf{pt} = \mathsf{D}(\mathsf{ct}, \mathsf{sk})$. Then, broadcast pt to all parties.

1. First the simulator emulates the ideal functionality $\mathcal{F}_{\mathsf{TAHE}}$ Setup command by randomizing a tuple $(\mathsf{pk}_{\mathsf{TAHE}}, \mathsf{sk}_{\mathsf{TAHE}})$ if the adversary aborts it samples a new value. Since the distribution of these public key is indistinguishable from this send by the functionality eventually either this process terminates or all corrupted parties are discovered.

2. $\mathcal{S}$ emulates the Key Generation phase via $\mathcal{F}_{\mathsf{DKG}}$.
3. For the pre-proccess phase the simulator works as follows:
   (a) the simulator calls the ideal functionality to get an encryption of $\mathsf{ct}_{\mathcal{F}}$.
   (b) The simulator computes $\widehat{\mathsf{ct}}^{i,\tau}$ and $\mathsf{ct}^{i,\tau}$ for every $i \notin U$ similarly to honest parties in the protocol. For every honest party it then sens $\mathsf{ct}^{i,\tau} + \mathsf{ct}_{\mathcal{F}}$ and $\widehat{\mathsf{ct}}^{i,\tau}$ along with a fake proof.
   (c) Upon receiving a proof Π_i^{zero} it verifies it and consider i malicious upon failure.
   (d) Upon receiving from the adversary a subset S of size $t+1$ which have sent valid messages. It uses $\mathsf{sk}_{\mathsf{TAHE}}$ to decrypt the messages $\widehat{\mathsf{ct}}^{\tau,i}$ sent by malicious parties and sums over all values in the subset S (which may include honest and malicious parties). To achieve the summed randomizer u_ℓ. This u_ℓ is to be used with the ℓ ciphertext which is being procced.
4. $\mathcal{S}$ emulates the threshold decryption:
   (a) Upon receiving a ciphertext $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)_\ell$, the simulator derives ciphertexts ct using key-switch similar to the real execution.
   (b) $\mathcal{S}$ sends $(\mathbf{decrypt}, \mathsf{ssid}, \mathsf{ct}, P_i, |S/U|)$ to $\mathcal{F}_{\texttt{ThFHE}}$ for every party. It then receives $u'_\ell = u_{\mathsf{ks}} + |S/U| u^\ell_{\mathcal{F}}$ where u_{ks} is the subset sum of the key-switch randomizer with the ciphertext and $u^\ell_{\mathcal{F}}$ is the randomizer of ct_ℓ.
   (c) The simulator emulates the threshold decryption part of $\mathcal{F}'_{\mathsf{TAHE}}$ to land on the value $u_\ell + u'_\ell$

In the following, we prove that the ideal and real executions are indistinguishable.

1. 1 - Indistinguishable since $\mathcal{F}_{\mathsf{TAHE}}$ has a UC simulator.
2. 2- Indistinguishable since $\mathcal{F}_{\mathsf{DKG}}$ has a UC simulator. Notice that at the end of the aforementioned simulation $\mathsf{ct}^{\mathsf{ks}}_u$ can be taken to be an encryption to zero.
3. 3-b indistinguishable from the RLWE assumption and the zk simulator of Π^{zero}.
4. 4-b Indeed the final ciphertext as randomizer $u_\ell + u_{\mathsf{ks}} + u^\ell_{\mathcal{F}}$ and thus upon opening it would fit ct_ℓ.
5. Indistinguishable since $\mathcal{F}_{\mathsf{TAHE}}$ has a UC simulator.

After decryption, the adversary $\mathcal{A}$ learns u'', which is the randomizer of ct''. However, u'' is masked by the aggregation of $t+1$ randomizers u_i, hiding the randomizer underlying the key switching ks. Thus, we can replace u'' with a random value.

In the real execution we will get randomizers of the form $u_{\mathsf{ks}} + u_\ell$ while in the real execution we will get $u_{\mathsf{ks}} + u_\ell + |S/U| u^\ell_{\mathcal{F}}$. we have that $u^\ell_{\mathcal{F}}$ statistically cover u_{ks} so for indistinguishability we need u_ℓ to statistically cover $u^\ell_{\mathcal{F}}$.

F Supporting Torus-FHE

F.1 Notation, Torus, and Polynomials over Torus

The real torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ represents the set of real numbers modulo 1. (i.e,. $\mathbb{T}$ consists of real numbers wrapped around a unit interval). Consider the polynomial rings $\mathbb{R}_N[X] = \mathbb{R}[X]/(X^N + 1)$ and $\mathbb{Z}_N[X] = \mathbb{Z}[X]/(X^N + 1)$, we have the $\mathbb{Z}_N[X]$-module $\mathbb{T}_N[X] = \mathbb{R}_N[X]/\mathbb{Z}_N[X] = \mathbb{T}[X]/(X^N + 1)$. Elements within $\mathbb{T}_N[X]$ can be viewed as polynomials modulo $X^N + 1$ with coefficients belonging to $\mathbb{T}$. Acting as a $\mathbb{Z}_N[X]$-module, elements in $\mathbb{T}_N[X]$ can be added together and externally multiplied by polynomials from $\mathbb{Z}_N[X]$. Mathematically, $\mathbb{T}$ is endowed with a $\mathbb{Z}$-module structure, reflecting its relationship with the integers. Let $\mathbb{B}$ be the integer subset $\{0, 1\}$ and, for N a power of $2, \mathbb{B}_N[X]$ is the subset of polynomials in $\mathbb{Z}_N[X]$ with coefficients in $\mathbb{B}$.

Any two elements of $\mathbb{T}$ can be added modulo 1, forming an abelian group denoted as $(\mathbb{T}, +)$. However, $\mathbb{T}$ does not constitute a ring due to the absence of a defined internal product $\times$ among its elements. Nevertheless, an external product denoted as $\cdot$ exists between integers and torus elements. For any $k \in \mathbb{Z}$ and $t \in \mathbb{T}$, the element $k \cdot t \in \mathbb{T}$ is defined as follows: $k \cdot t = t + \cdots + t$ (repeated $|k|$ times) if $k \geq 0$, and $k \cdot t = (-k) \cdot (-t)$ if $k < 0$. This definition ensures that the external product $k \cdot t$ aligns with the torus structure. The reverse negative wrapped convolution of two vectors $\mathbf{u} = (u_1, \ldots, u_n), \boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{Z}^{\mathrm{n}}$ is the vector $\boldsymbol{w} = \mathbf{u} \circledast \boldsymbol{v} = (\mathbf{u} \circledast_1 \boldsymbol{v}, \ldots, \mathbf{u} \circledast_{\mathrm{n}} \boldsymbol{v}) \in \mathbb{Z}^{\mathrm{n}}$ defined by

$$w_i = \mathbf{u} \circledast_i \boldsymbol{v} = \sum_{j=1}^{i} u_j v_{n+j-i} - \sum_{j=i+1}^{n} u_j v_{j-i}.$$

Definition 8. *[CGGI16] (LWE problem over the torus). Let $n \in \mathbb{N}$ and let $\mathbf{s} = (s_1, \ldots, s_n) \leftarrow \mathbb{B}^n$. Additionally, let χ represent an error distribution over $\mathbb{R}$. The learning with errors (LWE) over the torus problem involves distinguishing between the following distributions:*

- $\mathcal{D}_0 = \{(\mathbf{a}, r) \mid a \leftarrow \mathbb{T}^n, r \leftarrow \mathbb{T}\}$;
- $\mathcal{D}_1 = \Big\{(\mathbf{a}, r) \mid \mathbf{a} = (a_1, \ldots, a_n) \leftarrow \mathbb{T}^n, r = \sum_{j=1}^{n} s_j \cdot a_j + \ e, e \leftarrow \chi\}$.

Definition 9. *[CGGI16] (GLWE problem over the torus). Let $N, k \in \mathbb{N}$ with N a power of 2 and let $\boldsymbol{s} = (\jmath_1, \ldots, \jmath_k) \leftarrow \mathbb{B}_N[X]^k$. Additionally, let χ represent an error distribution over $\mathbb{R}_N[X]$. The general learning with errors (GLWE) over the torus problem involves distinguishing between the following distributions:*

- $\mathcal{D}_0 = \big\{(\boldsymbol{a}, r) \mid \mathbf{a} \leftarrow \mathbb{T}_N[X]^k, r \leftarrow \mathbb{T}_N[X]\big\}$;
- $\mathcal{D}_1 = \Big\{(\boldsymbol{a}, r) \mid \boldsymbol{a} = (a_1, \ldots, a_k) \leftarrow \mathbb{T}_N[X]^k, r = \sum_{j=1}^{k} s_j \cdot a_j + e, e \leftarrow \chi\Big\}$

The decisional LWE assumption (or the decisional GLWE assumption) asserts that it is computationally infeasible to solve the LWE problem (or GLWE problem) for a certain security parameter λ, where $n = n(\lambda)$ and $\chi = \chi(\lambda)$ (or $N = N(\lambda)$, $k = k(\lambda)$, and $\chi = \chi(\lambda)$).

PARAMETERS: An integer $n = 2^\eta$ for some $\eta > 0$, two positive integers p and q with $p \mid q$, let $\Delta = \mathrm{q/p}$, two discretized error distributions χ_1 and χ_2 from $\mathcal{N}\left(0, \sigma^2\right)$ over $\mathbb{Z}$. The plaintext space is $\mathcal{M} = \{0, 1, \ldots, t-1\}$. The public parameters are $\mathrm{pp} = \{n, \sigma, \mathrm{t}, \mathrm{q}, \Delta\}$

keygen$(2^\kappa) \to (\mathsf{sk}, \mathsf{pk})$.
- Sample uniformly at random a vector $\mathbf{s} = (s_1, \ldots, s_n) \leftarrow \{0,1\}^n$.
- Sample uniformly at random a vector $\mathbf{a} \leftarrow (\mathbb{Z}/\mathrm{q}\mathbb{Z})^n$ and compute $\mathbf{b} = \mathbf{a} \circledast \mathbf{s} + \boldsymbol{e} \in (\mathbb{Z}/\mathrm{q}\mathbb{Z})^n$ with $e \leftarrow {\chi_1}^n$.
- Output $(\mathsf{sk}, \mathsf{pk},)$ where $\mathsf{pk} = (\mathbf{a}, \mathbf{b})$, and $\mathsf{sk} = \mathbf{s}$.

$\mathsf{E}(\mathsf{pk}, m) \to \mathsf{ct}$
- Sample $\mathbf{r} \leftarrow \{0,1\}^n$, and $\boldsymbol{e}_1 \leftarrow \hat{\chi}_1{}^n$ and $e_2 \leftarrow \hat{\chi}_2$.
- Compute $c_0 = \mathbf{a} \circledast \mathbf{r} + e_1$ and $c_1 = \langle \mathbf{b}, \mathbf{r} \rangle + \Delta \mathrm{m} + \mathrm{e}_2$
- Output $\mathsf{ct} = (c_0, c_1)$

$\mathsf{D}(\mathsf{sk}, \mathsf{ct}) \to m$
- Let $c_0 = \mathsf{ct}[0], c_1 = \mathsf{ct}[1]$
- Compute $m^\star = c_1 - \langle \mathbf{c_0}, \mathbf{s} \rangle$
- Output $\lceil (\mu^* \bmod q) / \Delta \rfloor \bmod t$.

Fig. 8: A Public-key Torus FHE Construction from RLWE [CGGI16, Joy23].

F.2 Torus-FHE Construction and Its Extension for ThFHE

The LWE assumption over the torus essentially posits that a torus element $r \in \mathbb{T}$, constructed as $r = \sum_{j=1}^n s_j \cdot a_j + e$, cannot be distinguished from a random torus element $r \in \mathbb{T}$, even when the torus vector $(a_1, \ldots, a_n)$ is known. This $r = \sum_{j=1}^n s_j \cdot a_j + e$ can thus serve as a random mask to conceal a "plaintext message" $m \in \mathbb{T}$, forming a ciphertext $\boldsymbol{c} = (a_1, \ldots, a_n, r + m) \in \mathbb{T}^{n+1}$, where $s = (s_1, \ldots, s_n) \in \mathbb{B}^n$ acts as the private encryption key. In accordance with Protocol 1, we present the public-key Torus FHE in Figure 8.

With our DKG, integrating the key generation process of the traditional Torus-FHE becomes straightforward. Furthermore, we can customize our usage of VSS and ZKP to prevent malicious adversaries in the partial decryption. The remaining question pertains to the correctness of the resulting protocol as Torus-FHE operates with a small ciphertext modulus, while our asynchronous decryption requires a significantly larger modulus value. To tackle this, we propose employing the modulus switch operation [DDEK+23], enabling us to switch the ciphertext modulus to a desired one for successful decryption with our denominator clearing factor.