On Threshold Fully Homomorphic Encryption with Synchronized Decryptors

François Colin de Verdière[1], Alain Passelègue[2], and Damien Stehlé[2]

[1] ENS de Lyon, Lyon, France
[2] CryptoLab Inc., Lyon, France

Abstract. We study t-out-of-n threshold fully homomorphic encryption (ThFHE) in the synchronous setting, i.e., when the set of t decryptors is known at the outset of the decryption protocol. It has been observed in various works that the synchronous setting assumption enables to efficiently circumvent one of the major difficulties of ThFHE, namely hiding noise terms whose gigantic magnitude is incurred by the reconstruction coefficients of Shamir secret sharing. Yet, prior to this work, ThFHE in the synchronous setting had not even been properly defined.
As a first contribution, we provide a clear syntax for ThFHE in the synchronous setting, as well as two security models: a weak model in which designating the set of decryptors is only viewed as an enabler of efficient decryption and it is fine for non-designated parties to be able to decrypt as long as there are sufficiently many of them; and a much stronger model where a set of decryptors which does not exactly match the designated set should not learn anything. Then, we show that the main existing schemes (Mouchet et al., Journal of Cryptology, 2023; Mouchet et al., ACM CCS, 2024) are insecure, even in the weak model, by describing efficient key-recovery attacks. Finally, we propose a simple and efficient construction which achieves security in the strong security model, by masking the partial decryption shares with pseudorandom functions.

1 Introduction

Threshold fully homomorphic encryption [AJLA+12, BGG+18] (ThFHE) is a generalization of fully homomorphic encryption (FHE) [Gen09] to the distributed setting. In a ThFHE scheme, the decryption key is distributed between n parties such that any subset of t of them can decrypt, while any subset of at most $t-1$ of them learns nothing about the plaintexts corresponding to available ciphertexts. ThFHE is one of the most exciting extensions of FHE: it enables secure key management for FHE by distributing the secret key across several devices, it offers a simple solution to multiparty computation by running the computation homomorphically over the encrypted private data provided by the parties, and it provides a generic approach to thresholdize cryptographic algorithms [BGG+18].

To fix notation, recall that FHE schemes have the following structure (we focus on schemes based on Ring-LWE). The secret key is a ring element $\mathsf{sk} \in \mathcal{R}_q$,

where $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$ for N a power of 2 and $q \geq 2$, and a ciphertext is a pair $\mathsf{ct} = (a, b)$ satisfying $a \cdot \mathsf{sk} + b = \mathsf{Encode}(\mu) + e$, where $\mathsf{Encode}(\mu)$ is an encoding of the underlying message μ and e is a small error term with $\|e\| \leq B_{\mathsf{Dec}}$ for some bound $B_{\mathsf{Dec}} \ll q$. Decrypting a ciphertext ct is then done by applying the linear operation $(a, b) \mapsto a \cdot \mathsf{sk} + b$ and performing some post-processing (e.g., rounding and then applying Decode, the inverse of Encode).

The common approach for ThFHE design is to combine an FHE scheme of the above form together with a t-out-of-n linear secret sharing scheme. Linear secret sharing allows one to share a secret sk among n parties by distributing shares $\mathsf{sh}_1, \ldots, \mathsf{sh}_n$ such that any subset of t shares can reconstruct the secret sk as a linear combination of the t shares, where the linear coefficients of the combination depend on the subset. More explicitly, for any subset $\mathcal{S} = \{i_1, \ldots, i_t\} \subseteq [n]$ of t parties, there exist (efficiently computable) coefficients $\lambda_{i_1}^{\mathcal{S}}, \ldots, \lambda_{i_t}^{\mathcal{S}}$ such that:

$$\mathsf{sk} = \sum_{1 \leq j \leq t} \lambda_{i_j}^{\mathcal{S}} \cdot \mathsf{sh}_{i_j} \ .$$

Combining FHE with a linear secret sharing scheme yields the following ThFHE construction: the secret key sk is distributed between parties by providing a share sh_i to each party P_i, and distributed decryption is done by having each party output a partial decryption $a \cdot \mathsf{sh}_i$, such that $a \cdot \mathsf{sk}$ can be reconstructed by linearly combining any set of t distinct such partial decryptions. Unfortunately, revealing $a \cdot \mathsf{sh}_i$ directly reveals sh_i, and therefore this is insecure. To address this issue, the classical solution is to require the decryptor to add some error e_i^{smdg}, called the smudging error, in its partial decryption so that it instead reveals $a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}}$. Any subset of t partial decryption shares can then be combined to reconstruct $a \cdot \mathsf{sk}$ up to some error, as follows:

$$\sum_{1 \leq j \leq t} \lambda_{i_j}^{\mathcal{S}} \cdot (a \cdot \mathsf{sh}_{i_j} + e_{i_j}^{\mathsf{smdg}}) = a \cdot \mathsf{sk} + e^{\mathsf{smdg}} \ ,$$

where the error e^{smdg} stems from the linear combination of the individual smudging errors e_i^{smdg}'s, i.e., satisfies $e^{\mathsf{smdg}} = \sum_{1 \leq j \leq t} \lambda_{i_j}^{\mathcal{S}} \cdot e_{i_j}^{\mathsf{smdg}}$. Combining this with b, a user can then recover $a \cdot \mathsf{sk} + e^{\mathsf{smdg}} + b = \mathsf{Encode}(\mu) + e + e^{\mathsf{smdg}}$, which still provides μ if $e + e^{\mathsf{smdg}}$ is sufficiently small. In short, adding the e_i^{smdg}'s is necessary for security, while ensuring that e^{smdg} remains small provides correctness.

Unfortunately, it is not easy to enforce the smallness of e^{smdg} with the approach above. Two main solutions have been put forward in [BGG+18]. The first solution is to rely on Shamir secret sharing [Sha79] as the underlying t-out-of-n linear secret sharing scheme. The main benefit of this approach is that each share sh_i is a single $\mathcal{R}_q$-element. However, the reconstruction coefficients are (mod-q) fractions that can be as large as q, thereby preventing decryption. Boneh et al. proposed to clear out the denominators of the reconstruction coefficients by adding a scaled error $D \cdot e_i^{\mathsf{smdg}}$ so that the error e^{smdg} is small over the integers. Implementing this approach leads to a final error e^{smdg} that can be quite significant, and hence requires to choose a large modulus q. The cost can be daunting

as the number n of parties grows: in [BGG+18], the bit-size of q needs to grow as $\Omega(\lambda + n \log n)$ where λ refers to the security parameter (see [CKL24] for a discussion on the constant term). In [BFM+25], this was reduced to $\Omega(\lambda + n)$ by carefully choosing the Shamir secret sharing distinguished points. In concrete terms, this is still in the hundreds of bits even for moderate values of n. This still represents a massive overhead compared to single-party FHE. The second solution from [BGG+18] is to rely on a linear secret sharing scheme whose reconstruction coefficients are small, e.g., in $\{0, 1\}$. Such schemes exist and ensure correctness as e^{smdg} now remains small, but the shares sh_i's are now very large: the most compact such schemes distribute $O(n^{4.2})$ ring elements per party. As a result, neither solution allows one to construct a t-out-of-n ThFHE scheme with growing t and n, whose performance comes close to that of single-party FHE.

So far, we focused on the *asynchronous* setting, in which any set of t partial decryptions should allow decryption without knowing in advance which t decryptors will contribute. This is in contrast with the *synchronous* setting, in which the set of t decryptors is known at the outset of the decryption protocol. A brief remark about the potential efficiency advantage of considering this setting is made in [BGG+18, Footnote 1], though without an explicit scheme description. The synchronous setting *seems* to provide a direct solution to the issue of errors growing due to the reconstruction coefficients. Indeed, decryptors can directly reveal $\lambda_i^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}}$ instead of $a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}}$. By incorporating the reconstruction coefficient directly in the partial decryption, one prevents the decryption error from mingling with these coefficients, as combining partial decryption shares can now be achieved by adding them, leading to an added error $e^{\mathsf{smdg}} = \sum_{1 \leq j \leq t} e_{i_j}^{\mathsf{smdg}}$. This provides huge efficiency gains compared to constructions in the asynchronous setting: roughly $\lambda/2 + O(\log n)$ extra bits dedicated to the added error suffice to ensure both security and correctness. This led some practical works to focus on this setting, notably [MBH23, MCPT24]. While these works propose concrete constructions and protocols in the synchronous setting, they do not explicitly propose formal (functionality and security) definitions nor provide detailed security analyses in this setting. Yet, the fact that the set of decryptors is known at the time of decryption requires a modification in the definition of ThFHE: the (partial) decryption algorithm should take as an additional input the set of decryptors $\mathcal{S}$ for which it is crafting the partial decryption. Furthermore, this syntactic change to the partial decryption algorithm should be reflected in the security model. For instance, an attacker should be allowed to make decryption queries for different sets of decryptors for the same party. This is essential for practical deployment, as a target set of decryptors might fail to decrypt, e.g., if one of the decryptors is disconnected from the network before revealing its share. This is referred to as *churn* in [MCPT24]. In such a scenario, decryptors will attempt to decrypt with a different set of decryptors by replacing the now offline party and will reveal two partial decryptions of the same ciphertext for different target sets.

1.1 Contributions

We provide a formal treatement of ThFHE in the synchronous setting, showing that prior constructions are insecure and providing a secure and efficient solution.

In more details, our contributions are threefold. (1) First, we fill the definitional gap by providing a specific syntax for ThFHE in the synchronous setting, as well as two security models: a weak model in which designating the set of decryptors is only viewed as an enabler of efficient decryption and it is fine for non-designated parties to be able to decrypt as long as there are sufficiently many of them; and a much stronger model where a set of decryptors which does not exactly match the designated set should not learn anything. (2) Next, we describe efficient key-recovery attacks against the schemes proposed in [MBH23, MCPT24] in the weak security model. In particular, the simple scheme described above *is not secure*. It also implies that the scheme proposed in [MCPT24] does not provide security against churns as claimed. (3) Finally, we provide a simple and efficient construction which we prove secure in the strong security model. This construction achieves performance essentially matching the prior (insecure) constructions and relies on combining an FHE scheme with Shamir secret sharing and pseudorandom functions.

1.2 Technical Overview

We now provide a brief overview of our contributions, starting with syntax and security models for ThFHE in the synchronous setting.

ThFHE in the synchronous setting. We extend the syntax of ThFHE to the synchronous setting by adding the set of target decryptors as an additional input to the partial decryption algorithm. Hence, we define a *target-decryptors ThFHE* as a tuple of PPT algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$, with the following syntax. We focus on $\mathsf{KeyGen}, \mathsf{PartDec}$, and FinDec, as Enc and Eval have the same syntax as for classical FHE.

- $\mathsf{KeyGen}(1^\lambda, 1^n, 1^t)$ generates the public key pk together with partial decryption keys $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$;
- $\mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$ takes as input a partial decryption key sk_i, a ciphertext ct, and *a target set of decryptors* $\mathcal{S} \subseteq [n]$ *of size* t, and returns a partial decryption share $\mathsf{p}_i^{\mathcal{S}}$;
- $\mathsf{FinDec}((\mathsf{p}_i^{\mathcal{S}})_{i\in\mathcal{S}})$ takes as input t partial decryption shares, and returns a message μ.

For correctness, we ask that decryption succeeds when FinDec is fed with the t partial decryption shares $(\mathsf{p}_{i_1}^{\mathcal{S}}, \ldots, \mathsf{p}_{i_t}^{\mathcal{S}})$ corresponding to decryptors $(i_1, \ldots, i_t)$ in the same set $\mathcal{S}$.

A definition for ThFHE in the synchronous setting is proposed in [GKS24], following the same syntax as above, but only providing a very weak security notion (see discussion in Section 1.3). Our main definitional contribution is to propose two models of security for this syntax: a first, weaker notion which

known constructions already fail to satisfy, and a much stronger one, for which we propose a secure construction.

Our weak security notion, termed wk-IND-CPA-security, follows a classical indistinguishability-based game for ThFHE. The adversary can corrupt up to $t-1$ parties (i.e., learn up to $t-1$ partial decryption keys) and request challenge encryptions from an oracle by providing a pair of messages (μ_0, μ_1), to which the challenger replies by encrypting μ_b for a fixed unknown challenge bit $b \in \{0,1\}$. The adversary can also make evaluation queries with functions of its choice on previously obtained ciphertexts. In addition, it is given access to a partial decryption oracle, which it can query on input of a previously obtained ciphertext ct, an index $i \in [n]$, and a target set of decryptors $\mathcal{S}$. The challenger returns $\mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$ only if the plaintext underlying ct does not depend on the challenge bit b; otherwise the challenger returns $\perp$. This model is weak for the synchronous setting as it provides no confidentiality guarantee about the message underlying a ciphertext ct as soon as the attacker knows t partial decryptions of ct from distinct parties. In particular, there is no confidentiality guarantee even if the t partial decryptions are crafted for pairwise disjoint sets of decryptors. Yet, in the synchronous setting, it is entirely possible that a partial decryption for a target set of decryptors $\mathcal{S}$ is revealed and that it is still not possible to complete decryption, e.g., if the other decryptors in $\mathcal{S}$ are not corrupted and never reveal their shares.

We also propose a strong simulation-based security notion, termed stg-SIM security, which requires that corrupted partial decryption keys as well as partial decryption queries made by an attacker corrupting up to $t-1$ parties, can be efficiently simulated from the minimal information we intend the attacker to have. Specifically, corrupted partial decryption keys can be simulated given only the public key (and therefore reveal nothing about the secret key), and partial decryption shares $\mathsf{p}_i^{\mathcal{S}}$ of a ciphertext ct for an uncorrupted party P_i and a target set of decryptors $\mathcal{S}$ can be simulated: (1) from the public parameters if the attacker cannot decrypt ct given $\mathsf{p}_i^{\mathcal{S}}$ (i.e., if it is still missing some shares $\mathsf{p}_\ell^{\mathcal{S}}$ for $\ell \in \mathcal{S}$), or (2) from the additional knowledge of the plaintext μ underlying ct if and only if $\mathsf{p}_i^{\mathcal{S}}$ was the last missing partial decryption share for the attacker to decrypt ct using partial decryption shares from all decryptors in $\mathcal{S}$. Stated differently, the message μ is needed for simulating $\mathsf{p}_i^{\mathcal{S}}$ if and only, if for all $\ell \in \mathcal{S} \setminus \{i\}$, either P_ℓ is corrupted or the attacker already knows a partial decryption $\mathsf{p}_\ell^{\mathcal{S}}$ of ct.

This latter security notion is much stronger, as it guarantees that no information about the underlying plaintext leaks to an attacker as long as it has not collected t partial decryption shares for a target set of decryptors $\mathcal{S}$ from all parties in $\mathcal{S}$ (either by corrupting them or by submitting a decryption query). It notably guarantees security in the context of churns, as partial decryption shares for distinct sets of decryptors can be considered independent. Notably, our security notion guarantees that partial decryption shares of a ciphertext ct for a target set of decryptors $\mathcal{S}$ can be simulated independently of the knowledge of the underlying plaintext μ as long as the attacker cannot decrypt for $\mathcal{S}$, even

if ct has already been successfully decrypted by the attacker using a different set of decryptors $\mathcal{S}'$.

Attacks on prior schemes. We now explain why the prior constructions from [MBH23, MCPT24] are not wk-IND-CPA-secure. To do so, we exhibit key-recovery attacks that fit in the wk-IND-CPA model.

In [MBH23], the authors rely on the above idea of using Shamir secret sharing in the synchronous setting to obtain an efficient ThFHE scheme. Using the same notation as above, a partial decryption of a ciphertext $\mathsf{ct} = (a, b)$ is of the form

$$\mathsf{p}_i^{\mathcal{S}} = \lambda_i^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}} \ ,$$

where e_i^{smdg} is the smudging error and $\lambda_i^{\mathcal{S}}$ is a Lagrange coefficient of the form $\lambda_i^{\mathcal{S}} = \prod_{j \in \mathcal{S} \setminus \{i\}} \frac{\alpha_j}{\alpha_i - \alpha_j}$ where α_j is the Lagrange interpolation point of party P_j.

The adversary chooses a single ciphertext ct and queries partial decryptions for the same uncorrupted user, e.g., party P_1, for multiple sets $\mathcal{S}$ differing by one party, with the goal of recovering sh_1. Pick two sets of decryptors with $t-1$ common parties, i.e., $\mathcal{S}_0 = \mathcal{S}^* \cup \{i_0\}$ and $\mathcal{S}_1 = \mathcal{S}^* \cup \{i_1\}$ for some $\mathcal{S}^*$ of size $t-1$, and assume $1 \in \mathcal{S}^*$. Then we have:

$$\frac{\lambda_1^{\mathcal{S}_0}}{\lambda_1^{\mathcal{S}_1}} = \frac{\alpha_{i_0}(\alpha_1 - \alpha_{i_1})}{\alpha_{i_1}(\alpha_1 - \alpha_{i_0})} = \frac{\beta_1}{\gamma_1} \ .$$

By choosing interpolation points $\alpha_{i_0}, \alpha_{i_1}, \alpha_1$ appropriately, we can have $\beta_1, \gamma_1 \in \mathbb{Z}$, publicly computable, small, and with $\gcd(\beta_1, \gamma_1) = 1$. Then, querying $\mathsf{p}_1^{\mathcal{S}_0}$ and $\mathsf{p}_1^{\mathcal{S}_1}$, the attacker can compute:

$$\gamma_1 \cdot \mathsf{p}_1^{\mathcal{S}_0} - \beta_1 \cdot \mathsf{p}_1^{\mathcal{S}_1} = \gamma_1 e_0^{\mathsf{smdg}} - \beta_1 e_1^{\mathsf{smdg}} \bmod q \ ,$$

with $e_0^{\mathsf{smdg}}, e_1^{\mathsf{smdg}}$ being respectively the smudging errors in $\mathsf{p}_1^{\mathcal{S}_0}$ and $\mathsf{p}_1^{\mathcal{S}_1}$.

Note that the right-hand side contains only small elements. Therefore the equality holds over the integers, and reducing it modulo β_1 and multiplying by $\gamma_1^{-1} \bmod \beta_1$ gives $e_0^{\mathsf{smdg}} \bmod \beta_1$. Iterating over more sets $\mathcal{S}_2, \mathcal{S}_3, \ldots$ with $\mathcal{S}_j = \mathcal{S}^* \cup \{i_j\}$ for $j = 2, 3, \ldots$ then reveals $e_0^{\mathsf{smdg}} \bmod \beta_1, e_0^{\mathsf{smdg}} \bmod \beta_2, \ldots$, and the Chinese Remainder Theorem allows one to recover e_0^{smdg}, and therefore sh_1 (and then sk, since the adversary can corrupt $t-1$ other partial decryption keys).

The authors of [MBH23] noticed that the scheme could be insecure in the case of faulty decryptions (though without providing a concrete attack), and a subsequent work [MCPT24] explicitly mentions and tries to address potential security issues that arise in the presence of churns. This work proposes a modification of the scheme above and claims that it is secure, including in the presence of churns. The idea is to have decryptors of a ciphertext ct randomize it by adding a fresh encryption of 0 derived from the target set of decryptors $\mathcal{S}$, so that every attempt at decrypting ct with a different set of decryptors is actually a decryption of a distinct ciphertext. The hope is that this randomization breaks dependencies between partial decryptions of ct for different sets. The authors provide a high-level security analysis, which unfortunately does not hold.

We exhibit a key-recovery attack against this scheme, still in the wk-IND-CPA security model. Our attack follows a similar path to the attack above, though it is more convoluted due to the addition of the encryption of 0. What makes the analysis fail and the attack work is that, in order to ensure correctness of decryption, parties must generate the *same* encryption of 0 to randomize ct, as they must decrypt the *same* ciphertext to be able to combine partial decryptions. Therefore, parties (including the attacker) know the randomness used to generate the added encryption of 0, and knowing this randomness can be exploited to attack the scheme in a similar fashion as before. The strategy remains the same as above: combine several partial decryption shares for the same party but different well-chosen sets of decryptors in order to obtain equations over the integers, cancel out terms by keeping the remainder of a well-chosen Euclidean division, and rely on the Chinese Remainder Theorem to recover sh_1 given sufficiently many such remainders.

To complete the overview of our attacks, we emphasize that both attacks apply in particular to the most common instantiation of Shamir secret sharing, which shares sk coefficient-wise by sharing each of its coefficients over $\mathbb{Z}_q$ using Lagrange interpolation points i for party P_i, for all $i \in [n]$.

A secure construction. We finally describe our construction, which is a target-decryptors ThFHE that achieves the strong simulation security notion. It is obtained by tweaking the above basic construction that combines an FHE scheme with Shamir secret sharing. The tweak is inspired by the masking technique used in [dPKM+24] in the context of lattice-based threshold signatures and based on pseudorandom functions (PRF). We adapt the PRF masking to our setting as follows: during KeyGen, one further samples n^2 PRF keys $k_{i,j}$ for $i, j \in [n]$, and each party P_i receives $2n$ PRF keys $(k_{i,j}, k_{j,i})_{j\in[n]}$, in addition to its Shamir share sh_i of the FHE secret key sk. When a party P_i computes a partial decryption $\mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$, it computes a pseudorandom mask $r_i^{\mathcal{S},\mathsf{ct}} := \sum_{j\in\mathcal{S}}(F_{k_{i,j}}(\mathcal{S}, \mathsf{ct}) - F_{k_{j,i}}(\mathcal{S}, \mathsf{ct}))$ and uses it to hide its basic partial decryption share. Concretely, it reveals:

$$\mathsf{p}_i^{\mathcal{S}} = \lambda_i^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}} + r_i^{\mathcal{S},\mathsf{ct}} \ .$$

The key observations that make our scheme work are the following. First, for any subset $\mathcal{S}$ and any ciphertext ct, we have:

$$\sum_{i\in\mathcal{S}} r_i^{\mathcal{S},\mathsf{ct}} = \sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}(F_{k_{i,j}}(\mathcal{S}, \mathsf{ct}) - F_{k_{j,i}}(\mathcal{S}, \mathsf{ct})) = 0 \ .$$

This implies correctness, since pseudorandom masks cancel each other out when one combines t partial decryption shares for the same ciphertext ct and the same decryption set $\mathcal{S}$. Second, consider an attacker that corrupts a set $\mathcal{S}_{\mathsf{cor}}$ of (up to) $t-1$ parties and learns a partial decryption share of a ciphertext ct from an uncorrupted party P_i for a target set of decryptors $\mathcal{S}$. For simplicity, assume the adversary does not yet know any other partial decryption. Then one of the following holds.

- **Case 1:** $\mathcal{S} = \mathcal{S}_{\mathsf{cor}} \cup \{i\}$, i.e., the attacker can now decrypt ct and recover the underlying message μ. Then, by the observation above, we have:

$$r_i^{\mathcal{S},\mathsf{ct}} = -\sum_{j \in \mathcal{S}_{\mathsf{cor}}} \sum_{\ell \in \mathcal{S}} r_j^{\mathcal{S},\mathsf{ct}} \ .$$

  The right-hand side is computable by the attacker. One can therefore simulate the partial decryption share $\mathsf{p}_i^{\mathcal{S}}$ using the standard simulation technique based on noise smudging for ThFHE as if there were no masks (and just add the mask to correct the simulation).
- **Case 2:** $\mathcal{S} \neq \mathcal{S}_{\mathsf{cor}} \cup \{i\}$, i.e., there exists $\ell \in \mathcal{S} \setminus (\mathcal{S}_{\mathsf{cor}} \cup \{i\})$, and the attacker has no information about $k_{i,\ell}$ nor about $k_{\ell,i}$. In this situation, the mask $r_i^{\mathcal{S},\mathsf{ct}}$, which contains a term $F_{k_{i,\ell}}(\mathcal{S},\mathsf{ct}) - F_{k_{\ell,i}}(\mathcal{S},\mathsf{ct})$, is pseudorandom to the adversary, and $\mathsf{p}_i^{\mathcal{S}}$ can be simulated as a uniform value.

By extending this analysis, we prove that the above construction is secure. Note that its performance is essentially the same as prior (insecure) constructions. The overhead compared to prior work is that parties should store $2n$ PRF keys and perform $2t$ PRF evaluations during $\mathsf{PartDec}$. These are typically small compared to FHE evaluation keys and FHE evaluations, respectively.

In Appendix C, we argue that this scheme can be used to require $t' \gg t$ parties to participate in decryption, with security holding if fewer than t parties have been corrupted. This allows one to tune the number of parties required to participate in decryption, for example based on the sensitivity of the data underlying the ciphertext.

1.3 Related Works

A proper syntax and security definition for ThFHE in the synchronous setting is provided in [GKS24]. The syntax is the same as our, but their security notion is weak: multiple decryption queries on the same ciphertext or anything computed from the challenge is forbidden. The authors also propose a ThFHE scheme which matches that of [MBH23] and is subject to our attack (it is outside their model, hence not impacting their claims).

In [EY24], Ebrahimi and Yadav propose a stronger security notion for ThFHE in the asynchronous setting, compared to that from [BGG$^+$18]. It requires that the simulation of the partial decryption of a ciphertext should not require the knowledge of the underlying message if the adversary does not have enough partial shares to decrypt it. Hence, in some sense, our $\mathsf{stg\text{-}SIM}$ security notion can be seen as the synchronous version of the simulation security notion from [EY24]. Their strong security is achieved by combining the ThFHE scheme from [BGG$^+$18] with key-homomorphic pseudorandom functions [BLMR15] that serve as masks. We show in Appendix A that directly adapting their construction to the synchronous setting does not provide a secure construction (even for the weak security notion). Conversely, our PRF-based masking technique does not extend to the asynchronous setting from [EY24].

Our stg-SIM security notion posits that no information on the underlying plaintext can be obtained without obtaining t partial decryption shares for the same target decryptors set $\mathcal{S}$. This is reminiscent of the notion of context-dependent threshold public-key encryption introduced in [BBN+25], where partial decryption shares can be combined only if they are produced for the same context. Our stg-SIM security notion and our construction readily extend to context-dependent ThFHE in the synchronous setting. We refer the interested reader to Appendix B.

In [OT25], Okada and Takagi describe synchronous extensions of the asynchronous ThFHE from [PS24]. These works focus on decreasing the amount of noise smudging in partial decryptions, when the evaluator is not corrupted. The constructions from [OT25] rely on PRF-based masks to protect the decryption procedure (which in their case contains an additional pre-processing performed by the evaluator). Our technique presents important similarities to theirs but slightly differs: in their construction, the PRF keys are shared with the evaluator and the masks are added by the evaluator during the pre-processing of decryption and removed by decryptors during partial decryption. In addition, the authors also emphasize in [OT25, Section 2.3] the difficulty of achieving security for ThFHE in the synchronous setting and outline an attack which, assuming $q = \lambda_i^{\mathcal{S}} \cdot q'$ for some Lagrange coefficient $\lambda_i^{\mathcal{S}}$, allows an attacker to recover $a \cdot \mathsf{sh}_i \bmod q'$. Querying many such partial decryption allows to learn the lower bits of party P_i's share of the secret key $\mathsf{sh}_i \bmod q'$. The authors do not provide more details and we do not see how this translates into an actual attack against the scheme since information is modulo q'. In particular, a similar information is available in the asynchronous setting [BGG+18, CKL24], where a partial decryption has to form $a \cdot \mathsf{sh}_i + K \cdot e_i \bmod q$ for some (fixed) parameter K which serves as a denominator clearing factor for Lagrange coefficients. Then, assuming $q = K \cdot q'$, a partial decryption also reveals $a \cdot \mathsf{sh}_i \bmod K$, although the scheme is proven secure in [CKL24].

2 Preliminaries

Notation. Throughout this work, we let the security parameter be denoted by λ. We write $\mathsf{negl}(\lambda)$ for a negligible function in λ, i.e., a function $f = o(1/\lambda^c)$ for any integer $c > 0$. The notation log refers to the base-2 logarithm and SD denotes the statistical distance. For integers $0 < t \leq n$, we write $[n] = \{1, \ldots, n\}$ and $\mathbb{S}_{t,n} = \{\mathcal{S} \subseteq [n] \mid |\mathcal{S}| = t\}$. For integers $N, q > 0$ with N a power of 2, we define $\mathcal{R} = \mathbb{Z}[X]/(1+X^N)$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. If $e = \sum_{0 \leq i < N} e_i X^i$ is a polynomial in $\mathcal{R}$, we define its infinity norm $\|e\|_\infty = \max_{0 \leq i \leq N-1} |e_i|$. For a distribution $\mathcal{D}$, we let $x \leftarrow \mathcal{D}$ denote the action of sampling x from distribution $\mathcal{D}$. For a finite set $\mathcal{S}$, we let $U(\mathcal{S})$ denote the uniform distribution over $\mathcal{S}$. We write that an algorithm is PPT if it is probabilistic polynomial-time.

2.1 Smudging Lemma

We recall the following smudging lemma.

Lemma 2.1 (Adapted from [AJLA+12]). *Let B, B_{SM} be positive integers. For any $e \in \{-B, \ldots, B\}$, the distributions E_1 and $e + E_2$, where $E_1, E_2 \sim U(\{-B_{SM}, \ldots, B_{SM}\})$, satisfy:*

$$\mathrm{SD}\left(E_1, e + E_2\right) \leq \frac{2B}{B_{SM}} \ .$$

By extension, for a polynomial $e \in \mathcal{R}$ with $\|e\|_\infty \leq B$, and $E_1, E_2 \sim U(\mathcal{R}^{B_{SM}})$, where $\mathcal{R}^{B_{SM}}$ denotes the subset $\{E \in \mathcal{R} \mid \|E\|_\infty \leq B_{SM}\}$, we have:

$$\mathrm{SD}\left(E_1, e + E_2\right) \leq \frac{2NB}{B_{SM}} \ .$$

2.2 Pseudorandom Functions

In this section, we recall the definition of pseudorandom functions (PRFs). We follow a multi-key formulation as in [dPKM+24].

Definition 2.2 (PRFs). *A deterministic polynomial-time algorithm $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ with keyspace $\mathcal{K} = \mathcal{K}(\lambda)$, domain $\mathcal{X}$ and range $\mathcal{Y}$ is a pseudorandom function if, for any PPT adversary $\mathcal{A}$, we have:*

$$\left| \Pr[\mathsf{Exp}_F^{\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right| = \mathsf{negl}(\lambda) \ ,$$

where experiment $\mathsf{Exp}_F^{\mathcal{A}}$ is defined in Figure 1. We often abuse notation and let $F_k(x)$ denote $F(k, x)$.

The standard PRF security definition states that an adversary that does not know the PRF key cannot distinguish PRF outputs from uniform values. Definition 2.2 extends this to the multi-key setting: the adversary cannot distinguish PRF outputs from uniform values even when it can query evaluations under as many distinct, independent and uniform keys $(k_i)_i$ as it wishes. Multi-key security is implied by single-key security via a standard hybrid argument.

2.3 Shamir Secret Sharing

In this work, we use Shamir secret sharing over $R = \mathcal{R}_q$, and recall that Shamir secret sharing [Sha79] over a finite ring R allows one to share a secret in a t-out-of-n fashion via Lagrange interpolation. It relies on an exceptional set of $n + 1$ interpolation points $\alpha_0, \alpha_1, \ldots, \alpha_n \in R$, i.e., satisfying $\alpha_i - \alpha_j \in R^\times$ for all $1 \leq i \neq j \leq n$. As is done most often, we set $\alpha_0 = 0$ in this work.

For $t \leq n$, we let Share denote the algorithm that on input a number of parties n, a threshold t and a secret $s \in R$, returns secret shares $(\mathsf{sh}_1, \ldots, \mathsf{sh}_n) \leftarrow \mathsf{Share}(n, t, s)$. The algorithm proceeds as follows. It first samples a uniform degree-$(t-1)$ polynomial P over R satisfying $P(\alpha_0) = P(0) = s$, i.e., it samples $p_1, \ldots, p_{t-1} \leftarrow U(R)$ and defines $P(X) := s + \sum_{1 \leq i < t} p_i X^i$. Then, it returns $\mathsf{sh}_i = P(\alpha_i)$.

Algorithm 1: $\mathsf{Exp}_F^{\mathcal{A}}(\lambda)$

$b \leftarrow U(\{0,1\})$
$\mathsf{L}_k \leftarrow \emptyset$
$\mathsf{L}_q \leftarrow \emptyset$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_F^{\mathsf{PRF}}}$
return $b = b'$

$\mathcal{O}_F^{\mathsf{PRF}}(i, x)$

if $b = 0$ **then**
  if $\exists y : (i, x, y) \in \mathsf{L}_q$ **then**
    return y
  else
    $y \leftarrow U(\mathcal{Y})$
    $\mathsf{L}_q \leftarrow \mathsf{L}_q \cup \{(i, x, y)\}$
    return y
else
  if $\exists k : (i, k) \in \mathsf{L}_k$ **then**
    return $F(k, x)$
  else
    $k \leftarrow U(\mathcal{K})$
    $\mathsf{L}_k \leftarrow \mathsf{L}_k \cup \{(i, k)\}$
    return $F(k, x)$

Fig. 1. Security experiment for pseudorandom functions.

Any subset $\mathcal{S}$ of t parties can recover all the coefficients of the polynomial P using Lagrange interpolation, and thus the secret s as:

$$s = \sum_{i \in \mathcal{S}} \lambda_i^{\mathcal{S}} \cdot \mathsf{sh}_i, \quad \text{where} \quad \lambda_i^{\mathcal{S}} = \prod_{j \in \mathcal{S} \setminus \{i\}} \frac{\alpha_j}{\alpha_i - \alpha_j} \ . \tag{1}$$

The $(\lambda_i^{\mathcal{S}})$'s are the Lagrange coefficients for the points $(\alpha_i)_{i \in \mathcal{S}}$.

Privacy. Shamir secret sharing achieves information-theoretic privacy. Specifically, for any subset $\mathcal{S}_{\mathsf{cor}} \subseteq [n]$ with $|\mathcal{S}_{\mathsf{cor}}| < t$, for any $s_0, s_1 \in R$, define distributions $\mathcal{D}_b = \{(\mathsf{sh}_i)_{i \in \mathcal{S}_{\mathsf{cor}}} | (\mathsf{sh}_1, \ldots, \mathsf{sh}_n) \leftarrow \mathsf{Share}(n, t, s_b)\}$ for $b \in \{0, 1\}$. Then $\mathcal{D}_0$ and $\mathcal{D}_1$ are identically distributed.

2.4 Ring-LWE

We describe our contributions for FHE schemes that rely on the Ring-LWE problem [Bra12, FV12, BGV12, CKKS17]. They readily extend to schemes relying on similar problems, such as LWE [CGGI16].

The Ring-LWE assumption [SSTX09, LPR10] states that samples $(a_i, b_i = -a_i \cdot s + e_i)_{i \in [m]}$ are computationally indistinguishable from uniform over $(\mathcal{R}_q^2)^m$, when the a_i's are sampled uniformly in $\mathcal{R}_q$, and when s and the e_i's are sampled in $\mathcal{R}$ from a specified distribution that outputs samples that have infinity norms smaller than a bound $B \ll q$. The latter distribution, N, q and m are problem parameters.

The only assumptions we make on the underlying FHE scheme is that a ciphertext to be decrypted is of the form $(a, b) \in \mathcal{R}_q^2$ with $b = -a \cdot s + e + \mathsf{Encode}(\mu)$,

where s is the secret key, e has an infinity norm bounded by a known $B_{\mathsf{Dec}} \ll q$ and $\mathsf{Encode}(\mu)$ is an encoding of the message μ. The encoding function Encode has an inverse function Decode and both are efficiently computable.

3 ThFHE with Target Decryptors

The classical definition for ThFHE [BGG+18] does not capture the case where the decryptors synchronized to know who they are. Even in works specifically studying this ThFHE variant, no dedicated syntax and security definitions are provided (see [MBH23, MCPT24]). In this section, we fill this gap by providing a definition as well as two security models for ThFHE in the synchronous model.

3.1 Syntax

Our syntax for ThFHE with target decryptors extends the classical syntax for ThFHE (the latter can be recovered by removing the set of decryptors $\mathcal{S}$ from the $\mathsf{PartDec}$ inputs). We do not explicitly consider the evaluation key and simply consider it as part of the public key.

Definition 3.1 (Target-Decryptors ThFHE). *A target-decryptors ThFHE scheme is a tuple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$ *with the following specifications.*

- $\mathsf{KeyGen}(1^\lambda, 1^n, 1^t)$*: on input a security parameter* λ*, a number of parties* n*, and a threshold* $t \leq n$*, it outputs* n *secret-key shares* $(\mathsf{sk}_i)_{i\in[n]}$ *and a public key* pk*.*
- $\mathsf{Enc}(\mathsf{pk}, \mu)$*: on input a public key* pk *and a message* μ*, it outputs a ciphertext* ct*.*
- $\mathsf{Eval}(\mathsf{pk}, \mathsf{C}, (\mathsf{ct}_j)_{j\in[\ell]})$*: on input a public key* pk*, a circuit* C *with arity* $\ell \geq 1$ *and ciphertexts* $(\mathsf{ct}_j)_{j\in[\ell]}$*, it outputs a ciphertext* $\widehat{\mathsf{ct}}$*.*
- $\mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$*: on input a ciphertext* ct*, a secret-key share* sk_i*, and a set of decryptors* $\mathcal{S} \subseteq [n]$ *with* $|\mathcal{S}| = t$*, it outputs a partial decryption share* $\mathsf{p}_i^{\mathcal{S}}$*.*
- $\mathsf{FinDec}(\mathsf{ct}, (\mathsf{p}_i^{\mathcal{S}})_{i\in\mathcal{S}})$*: on input a ciphertext* ct *and* t *partial decryption shares* $(\mathsf{p}_i^{\mathcal{S}})_{i\in\mathcal{S}}$ *for a set of decryptors* $\mathcal{S}$*, it outputs a message* μ*.*

Correctness.[3] *For any circuit* C *with arity* $\ell \geq 1$*, for any messages* $(\mu_j)_{j\in[\ell]}$*, for any* n*, and any subset* $\mathcal{S}$ *of* t *parties, we require* $p \geq 1 - \mathsf{negl}(\lambda)$*, where:*

$$p = \Pr\left[\mathsf{FinDec}(\mathsf{ct}, (\mathsf{p}_i^{\mathcal{S}})_{i\in\mathcal{S}}) = \mathsf{C}((\mu_j)_{j\in[\ell]}) \;\middle|\; \begin{array}{c} ((\mathsf{sk}_i)_{i\in[n]}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^n, 1^t) \\ \mathsf{ct}_j \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu_j), \forall j \leq \ell \\ \mathsf{ct} \leftarrow \mathsf{Eval}(\mathsf{pk}, \mathsf{C}, (\mathsf{ct}_j)_{j\in[\ell]}) \\ \mathsf{p}_i^{\mathcal{S}} \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S}), \forall i \in \mathcal{S} \end{array}\right]$$

and the probability is taken over the randomness of the algorithms.

[3] For simplicity, we rely on perfect correctness, but our main construction extends to approximate schemes.

Compactness. *There exists a polynomial P such that for any λ, n, t, and execution of* $\mathsf{KeyGen}(1^\lambda, 1^n, 1^t)$*, all ciphertexts produced by any combination of* Enc *and* Eval *with arbitrary circuits* C *have bit-size at most* $P(\lambda, n)$*.*

3.2 Weak Security Model

As Definition 3.1 provides the set of decryptors $\mathcal{S}$ as input to $\mathsf{PartDec}$, the classical security definitions for ThFHE need to be adapted. We provide two possible extensions, depending on whether one considers that $\mathcal{S}$ is provided only to gain efficiency or also security. In the weak definition, using t decryption shares generated for different sets $\mathcal{S}$ to gain information on the underlying plaintext is not viewed as a valid attack: the set $\mathcal{S}$ is provided for convenience only and it is not an issue if unforeseen decryptors collaborate to decrypt as long as there are at least t of them. In the strong definition, the decryption shares can be combined only for the set $\mathcal{S}$ for which they were created: combining shares produced for different sets should not help the adversary gain information about the underlying plaintext. Notably, an adversary could collect more than t decryption shares for the same ciphertext and still not learn anything about the underlying message, as long as there are not t shares for a same decryptors set.

Below, we give an indistinguishability-based definition for the weak notion. It may be adapted to a simulation-based definition, but as we give attacks for the weak notion, we give the weaker of the two definitions. Note that, for this weak definition, multiple decryption shares given by the same party but for different sets of decryptors should not leak information about that party's secret key.

Definition 3.2 (Weak td-ThFHE Security). *A target-decryptors ThFHE scheme* $\mathsf{ThFHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$ *is said* $\mathsf{wk\text{-}IND\text{-}CPA}$*-secure if, for any PPT adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*, we have* $\delta \leq \mathsf{negl}(\lambda)$ *where:*

$$\delta = \left| \Pr[\mathcal{A}(\mathsf{Exp}_1^{\mathsf{wk\text{-}IND\text{-}CPA}}(1^\lambda, 1^n, 1^t)) = 1] - \Pr[\mathcal{A}(\mathsf{Exp}_0^{\mathsf{wk\text{-}IND\text{-}CPA}}(1^\lambda, 1^n, 1^t)) = 1] \right|,$$

for the experiments $(\mathsf{Exp}_b^{\mathsf{wk\text{-}IND\text{-}CPA}})_{b\in\{0,1\}}$ *described in Figure 2.*

3.3 Strong Security Model

We now focus on the strong security model, where decryption shares for the same ciphertext but different decryptor sets of size $< t$ should not reveal information about the underlying plaintext. We provide a simulation-based definition of the strong security model. It is also possible to give an indistinguishability-based definition, but as we focus on proving security, we present the strongest version. Our security notion is analogous to that proposed in [EY24] in the asynchronous setting, with ours being more permissive regarding adaptive queries. Note that the security model from [EY24] is stronger than the original one from [BGG$^+$18].

To start, let us note that according to our definition of target-decryptors ThFHE, a decryption share of a ciphertext ct is associated with two attributes:

$\mathsf{Exp}_b^{\mathsf{wk\text{-}IND\text{-}CPA}}(1^\lambda, 1^n, 1^t)$:

1 $((\mathsf{sk}_i)_{i\in[n]}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^n, 1^t)$
2 $\mathsf{ctr} \leftarrow 0,\ \mathsf{L} \leftarrow \emptyset$
3 $(\mathcal{S}_{\mathsf{cor}}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_0(\mathsf{pk})$ with $\mathcal{S}_{\mathsf{cor}} \subseteq [n]$ and $|\mathcal{S}_{\mathsf{cor}}| < t$
4 $b \leftarrow \mathcal{U}(\{0,1\})$
5 $b' \leftarrow \mathcal{A}_1^{\mathsf{OEnc},\mathsf{OChall}_b,\mathsf{OEval},\mathsf{ODec}}((\mathsf{sk}_i)_{i\in\mathcal{S}_{\mathsf{cor}}}, \mathsf{pk}, \mathsf{st}_{\mathcal{A}})$
6 **return** $(b' = b)$

$\mathsf{OEnc}(\mu)$:

1 $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu)$
2 $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
3 $\mathsf{L}[\mathsf{ctr}] \leftarrow (\mu, \mu, \mathsf{ct})$
4 **return** ct

$\mathsf{OEval}(\mathsf{C}, (j_1, \ldots, j_\ell))$:

1 **if** $\exists k \in [\ell] : j_k > \mathsf{ctr}$ **then**
2 **return** $\perp$
3 **for** $k \in [\ell]$ **do**
4 $(\mu_{0,k}, \mu_{1,k}, \mathsf{ct}_k) \leftarrow \mathsf{L}[j_k]$
5 $\mathsf{ct} \leftarrow \mathsf{Eval}(\mathsf{pk}, \mathsf{C}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell))$
6 $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
7 $\mu_0 \leftarrow \mathsf{C}(\mu_{0,1}, \ldots, \mu_{0,\ell})$
8 $\mu_1 \leftarrow \mathsf{C}(\mu_{1,1}, \ldots, \mu_{1,\ell})$
9 $\mathsf{L}[\mathsf{ctr}] \leftarrow (\mu_0, \mu_1, \mathsf{ct})$
10 **return** ct

$\mathsf{OChall}_b(\mu_0, \mu_1)$:

1 **if** $|\mu_0| \neq |\mu_1|$ **then**
2 **return** $\perp$
3 $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu_b)$
4 $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
5 $\mathsf{L}[\mathsf{ctr}] \leftarrow (\mu_0, \mu_1, \mathsf{ct})$
6 **return** ct

$\mathsf{ODec}(j, i, \mathcal{S})$:

1 **if** $j > \mathsf{ctr} \vee |\mathcal{S}| \neq t \vee i \notin \mathcal{S}$ **then**
2 **return** $\perp$
3 $(\mu_0, \mu_1, \mathsf{ct}) \leftarrow \mathsf{L}[j]$
4 **if** $\mu_0 \neq \mu_1$ **then**
5 **return** $\perp$
6 $\mathsf{p}_i^{\mathcal{S}} \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$
7 **return** $\mathsf{p}_i^{\mathcal{S}}$

Fig. 2. wk-IND-CPA security games for target-decryptors ThFHE.

the index $i \in [n]$ of the decryption key share sk_i used to generate it, as well as a set of decryptors $\mathcal{S} \subseteq [n]$ satisfying $i \in \mathcal{S}$ and $|\mathcal{S}| = t$ for which it is crafted. To simplify the description of our security model, we introduce an auxiliary algorithm CanCombine, described in Figure 3, which tells whether a PartDec query $(i, \mathcal{S})$ on a ciphertext ct and for a target set $\mathcal{S}$ allows the adversary to recover the underlying plaintext based on the information it already has (corrupted keys and prior partial decryption shares). Specifically, CanCombine takes as inputs the set (of indices) $\mathcal{S}_{\mathsf{cor}} \subseteq [n]$ of corrupted secret keys owned by the adversary, the set $\mathsf{P}[\mathsf{ct}]$ containing all prior partial decryption queries $(i_k, \mathcal{S}_k)$ made on ct by the adversary, and the new query $(i, \mathcal{S})$. Then CanCombine returns 1 if ct can be decrypted given the answer to query $(i, \mathcal{S})$ by combining it with other decryption shares, i.e., CanCombine returns 1 if $i \notin \mathcal{S}_{\mathsf{cor}}$ and for all $k \in \mathcal{S} \setminus \{i\}$, at least one of the following conditions holds:

- $k \in \mathcal{S}_{\mathsf{cor}}$, i.e., the adversary knows sk_k;

$\mathsf{Exp}^{\mathsf{stg\text{-}SIM}}_{\mathsf{real}}(1^\lambda, 1^n, 1^t)$:

$((\mathsf{sk}_i)_{i\in[n]}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^n, 1^t)$
$\mathsf{ctr} \leftarrow 0,\ \mathsf{L} \leftarrow \varnothing,\ \mathsf{P} \leftarrow \varnothing$
$(\mathcal{S}_{\mathsf{cor}}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_0(\mathsf{pk})$ with $|\mathcal{S}_{\mathsf{cor}}| < t$
$\mathsf{out} \leftarrow \mathcal{A}_1^{\mathsf{OEnc},\mathsf{OEval},\mathsf{ODec}}((\mathsf{sk}_i)_{i\in\mathcal{S}_{\mathsf{cor}}}, \mathsf{pk}, \mathsf{st}_{\mathcal{A}})$
return $(\mathsf{out} = \mathsf{real})$

$\mathsf{OEnc}(\mu)$:

$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu)$
$\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
if $\mathsf{P}[\mathsf{ct}]$ *does not exist* **then**
  $\mathsf{P}[\mathsf{ct}] \leftarrow \varnothing$
$\mathsf{L}[\mathsf{ctr}] \leftarrow (\mu, \mathsf{ct})$
return ct

$\mathsf{OEval}(\mathsf{C}, (j_1, \ldots, j_\ell))$:

if $\exists k \in [\ell] : j_k > \mathsf{ctr}$ **then**
  return $\perp$
for $k \in [\ell]$ **do**
  $(\mu_k, \mathsf{ct}_k) \leftarrow \mathsf{L}[j_k]$
$\mathsf{ct} \leftarrow \mathsf{Eval}(\mathsf{pk}, \mathsf{C}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell))$
$\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
$\mu \leftarrow \mathsf{C}(\mu_1, \ldots, \mu_\ell)$
if $\mathsf{P}[\mathsf{ct}]$ *does not exist* **then**
  $\mathsf{P}[\mathsf{ct}] \leftarrow \varnothing$
$\mathsf{L}[\mathsf{ctr}] \leftarrow (\mu, \mathsf{ct})$
return ct

$\mathsf{Exp}^{\mathsf{stg\text{-}SIM}}_{\mathsf{ideal}}(1^\lambda, 1^n, 1^t)$:

$((\mathsf{sk}_i)_{i\in[n]}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^n, 1^t)$
$\mathsf{ctr} \leftarrow 0,\ \mathsf{L} \leftarrow \varnothing,\ \mathsf{P} \leftarrow \varnothing$
$(\mathcal{S}_{\mathsf{cor}}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_0(\mathsf{pk})$ with $|\mathcal{S}_{\mathsf{cor}}| < t$
$(\mathsf{st}, (\mathsf{sk}'_i)_{i\in\mathcal{S}_{\mathsf{cor}}}) \leftarrow \mathsf{Sim}_0(\mathsf{pk}, \mathcal{S}_{\mathsf{cor}})$
$\mathsf{out} \leftarrow \mathcal{A}_1^{\mathsf{OEnc},\mathsf{OEval},\mathsf{OSim}}((\mathsf{sk}'_i)_{i\in\mathcal{S}_{\mathsf{cor}}}, \mathsf{pk}, \mathsf{st}_{\mathcal{A}})$
return $(\mathsf{out} = \mathsf{ideal})$

$\mathsf{ODec}(j, i, \mathcal{S})$:

if $j > \mathsf{ctr} \vee |\mathcal{S}| \neq t \vee i \notin \mathcal{S}$ **then**
  return $\perp$
$(\mu, \mathsf{ct}) \leftarrow \mathsf{L}[j]$
$\mathsf{p}_i^{\mathcal{S}} \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$
return $\mathsf{p}_i^{\mathcal{S}}$

$\mathsf{OSim}(j, i, \mathcal{S})$:

if $j > \mathsf{ctr} \vee |\mathcal{S}| \neq t \vee i \notin \mathcal{S}$ **then**
  return $\perp$
$(\mu, \mathsf{ct}) \leftarrow \mathsf{L}[j]$
if $\mathsf{CanCombine}(\mathcal{S}_{\mathsf{cor}}, \mathsf{P}[\mathsf{ct}], i, \mathcal{S})$ **then**
  $(\mathsf{st}, \mathsf{p}_i^{\mathcal{S}}) \leftarrow \mathsf{Sim}_2(\mathsf{st}, \mathsf{ct}, i, \mathcal{S}, \mu)$
else
  $(\mathsf{st}, \mathsf{p}_i^{\mathcal{S}}) \leftarrow \mathsf{Sim}_1(\mathsf{st}, \mathsf{ct}, i, \mathcal{S})$
$\mathsf{P}[\mathsf{ct}] \leftarrow \mathsf{P}[\mathsf{ct}] \cup \{(i, \mathcal{S})\}$
return $\mathsf{p}_i^{\mathcal{S}}$

$\mathsf{CanCombine}(\mathcal{S}_{\mathsf{cor}}, \mathsf{P}[\mathsf{ct}], i, \mathcal{S})$:

if $i \in \mathcal{S}_{\mathsf{cor}}$ **then**
  return 0
else
  if $\forall k \in \mathcal{S} \setminus \{i\}, k \in \mathcal{S}_{\mathsf{cor}} \vee (k, \mathcal{S}) \in \mathsf{P}[\mathsf{ct}]$ **then**
    return 1
  else
    return 0

Fig. 3. stg-SIM security games for target-decryptors ThFHE.

- $(k, \mathcal{S}) \in \mathsf{P}[\mathsf{ct}]$, i.e., the adversary knows a partial decryption of ct for party P_k and set $\mathcal{S}$.

By convention, we make $\mathsf{CanCombine}$ return 0 when $i \in \mathcal{S}_{\mathsf{cor}}$, as partial decryptions can always be computed by the attacker in this case and therefore the adversary does not learn anything new from this query.

At a high level, our security notion states the following. Consider an adversary that may corrupt up to $t-1$ parties. For a ciphertext ct encrypting a certain plaintext μ, we have two cases:

(1) The adversary has enough information to decrypt ct: in this case, the adversary should learn the underlying plaintext μ but nothing more;
(2) The adversary is not able to decrypt ct: in this case, the adversary should learn nothing.

We formalize this by relying on two simulators that can simulate partial decryption queries based on the view of the attacker and (1) the underlying plaintext, or (2) only the public parameters.

To further illustrate the strength of our definition, note that an adversary corrupting a set $\mathcal{S}_{\mathsf{cor}}$ of $t-1$ parties still learns nothing about the plaintext underlying a ciphertext ct even if it obtains a partial decryption share $\mathsf{p}_i^{\mathcal{S}}$ of ct from a non-corrupted party P_i with $i \notin \mathcal{S}_{\mathsf{cor}}$, as long as $\mathcal{S} \neq \mathcal{S}_{\mathsf{cor}} \cup \{i\}$.

Definition 3.3 (Strong td-ThFHE Security). *A target-decryptors ThFHE scheme* $\mathsf{ThFHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$ *is said* $\mathsf{stg\text{-}SIM}$*-secure if it satisfies the following two properties:*

- *the sub-scheme* $(\mathsf{KeyGen}, \mathsf{Enc})$ *is* $\mathsf{IND\text{-}CPA}$*-secure;*
- *there exists a PPT simulator* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1, \mathsf{Sim}_2)$ *such that the experiments depicted in Figure 3 are computationally indistinguishable; specifically, we require that for any PPT adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*, we have* $\delta \leq \mathsf{negl}(\lambda)$ *where*

$$\delta = \left| \Pr[\mathcal{A}(\mathsf{Exp}^{\mathsf{stg\text{-}SIM}}_{\mathsf{real}}(1^\lambda, 1^n, 1^t)) = 1] - \Pr[\mathcal{A}(\mathsf{Exp}^{\mathsf{stg\text{-}SIM}}_{\mathsf{ideal}}(1^\lambda, 1^n, 1^t)) = 1] \right|.$$

4 Insecurity of Prior Target-Decryptors ThFHE Schemes

In this section, we show that the target-decryptors ThFHE schemes described in [MBH23] and [MCPT24] are insecure. For each attack, we illustrate how it applies to the most common setup, i.e. when the secret key is shared coefficient-wise over $\mathbb{Z}_q$ via Shamir secret sharing with interpolations $\alpha_i = i$ for $i \in [n]$.

4.1 ThFHE Scheme from [MBH23]

The first explicit target-decryptors ThFHE candidate was given in [MBH23]. We prove that it is not secure even for the weak security model (wk-IND-CPA).

Scheme description. We first recall the scheme. The secret key is an element $\mathsf{sk} \in \mathcal{R}_q$ for some integer $q \geq 2$. It is secret-shared as $(\mathsf{sh}_i)_{i\in[n]}$ using the Shamir secret sharing scheme: given any set $\mathcal{S}$ of t parties, there exist linear reconstruction coefficients $(\lambda_i^{\mathcal{S}})_{i\in\mathcal{S}}$ as in Equation (1). The i-th party receives $\mathsf{sk}_i = \mathsf{sh}_i$ as its secret-key share. Now, a ciphertext for the plaintext $\mu \in \mathcal{R}$ is of the form $\mathsf{ct} = (a, b) \in \mathcal{R}_q^2$, with $b + a \cdot \mathsf{sk} = \Delta\mu + e \bmod q$, where $\|e\| \ll q$ and $\Delta \in \mathbb{Z}$ is a scaling factor. To decrypt ct for a target set $\mathcal{S}$, each party $i \in \mathcal{S}$ computes the decryption share $\mathsf{p}_i^{\mathcal{S}} = \lambda_i^{\mathcal{S}} a \cdot \mathsf{sh}_i + e_i$, where $\|e_i\|$ is a smudging noise that is $2^{\Omega(\lambda)}$-times larger than a known bound on $\|e\|$. Finally, given the decryption shares $(\mathsf{p}_i^{\mathcal{S}})_{i\in\mathcal{S}}$, FinDec makes the following computation modulo q:

$$\begin{aligned} b + \sum_{i\in\mathcal{S}} \mathsf{p}_i^{\mathcal{S}} &= b + a \sum_{i\in\mathcal{S}} \lambda_i^{\mathcal{S}} \mathsf{sh}_i + \sum_{i\in\mathcal{S}} e_i \\ &= b + a \cdot \mathsf{sk} + \sum_{i\in\mathcal{S}} e_i = \Delta\mu + e + \sum_{i\in\mathcal{S}} e_i \ . \end{aligned}$$

If the scheme parameters are set appropriately, the final error term $e + \sum_{i\in\mathcal{S}} e_i$ has a small magnitude compared to Δ, and $\Delta\mu + e + \sum_{i\in\mathcal{S}} e_i$ has a small magnitude compared to q. Assuming this is the case, the term $\Delta\mu + e + \sum_{i\in\mathcal{S}} e_i$ is known over $\mathcal{R}$ rather than merely $\mathcal{R}_q$, and μ can be recovered by rounding.

Cryptanalysis. We now describe our attack. It requires a single ciphertext ct for plaintext $\mu = 0$, and outputs the secret-key share $\mathsf{sk}_1 = \mathsf{sh}_1$ of the first user. Assuming $t - 1$ other users have been corrupted, the attacker can recover the secret key by using Equation (1) and hence decrypt any ciphertext. Write $\mathsf{ct} = (a, b)$ with $b + a \cdot \mathsf{sk} = e \bmod q$ and $\|e\| \ll q$.

We start from the observation that the ratio of two Lagrange coefficients can be very small. For the sake of simplicity, we assume that all reconstruction coefficients are invertible in $\mathcal{R}_q$ (using the remark at the end of Section 2.3, for example). For instance, consider two sets $\mathcal{S}_0$ and $\mathcal{S}_1$ with $|\mathcal{S}_0| = |\mathcal{S}_1| = t$ and $1 \in \mathcal{S}^* = \mathcal{S}_0 \cap \mathcal{S}_1$, and that differ in a single point, i.e.:

$$\mathcal{S}_0 = \mathcal{S}^* \cup \{i_0\} \quad \text{and} \quad \mathcal{S}_1 = \mathcal{S}^* \cup \{i_1\} \ .$$

Then, by Equation (1), the following holds modulo q:

$$\frac{\lambda_1^{\mathcal{S}_0}}{\lambda_1^{\mathcal{S}_1}} = \frac{\alpha_{i_0}(\alpha_1 - \alpha_{i_1})}{\alpha_{i_1}(\alpha_1 - \alpha_{i_0})} = \frac{\beta_1}{\gamma_1} \ ,$$

with $\beta_1 = \alpha_{i_0}(\alpha_1 - \alpha_{i_1})$ and $\gamma_1 = \alpha_{i_1}(\alpha_1 - \alpha_{i_0})$. If the distinguished points for Shamir secret sharing are small integers, then both $\alpha_{i_0}(\alpha_1 - \alpha_{i_1})$ and $\alpha_{i_1}(\alpha_1 - \alpha_{i_0})$ are small integers. We further assume that the Shamir secret sharing interpolation points are such that β_1 and γ_1 are coprime.

Based on this observation, the attacker requests a partial decryption of $\mathsf{ct} = (a, b)$ for the first user, for both target sets $\mathcal{S}_0$ and $\mathcal{S}_1$. It recovers the following elements of $\mathcal{R}_q$:

$$\mathsf{p}_1^{\mathcal{S}_0} = \lambda_1^{\mathcal{S}_0} a \cdot \mathsf{sh}_1 + e_0 \ \text{ and } \ \mathsf{p}_1^{\mathcal{S}_1} = \lambda_1^{\mathcal{S}_1} a \cdot \mathsf{sh}_1 + e_1 \ .$$

By construction of $\mathcal{S}_0$ and $\mathcal{S}_1$, we have, modulo q:

$$\mathsf{p}_1^{\mathcal{S}_0} \cdot \gamma_1 - \mathsf{p}_1^{\mathcal{S}_1} \cdot \beta_1 = e_0\gamma_1 - e_1\beta_1 \ .$$

The right-hand side has a small magnitude compared to q and is hence known over $\mathcal{R}$. Reducing it modulo β_1 and multiplying by the inverse of γ_1 modulo β_1 (recall that they are coprime) gives $e_0 \bmod \beta_1$.

The attacker performs the process above with $(\mathcal{S}_0, \mathcal{S}_1), (\mathcal{S}_0, \mathcal{S}_2), (\mathcal{S}_0, \mathcal{S}_3), \ldots$ for pairwise-distinct sets $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \ldots$ of size t, with $\mathcal{S}_k = \mathcal{S}^* \cup \{i_k\}$ for $k \geq 0$ and $1 \in \mathcal{S}^*$. This gives $e_0 \bmod \beta_1, e_0 \bmod \beta_2, e_0 \bmod \beta_3, \ldots$ for β_i's that can be made pairwise coprime. Using the Chinese Remainder Theorem, the attacker can recover $e_0 \in \mathcal{R}$ and then sh_1 from $\mathsf{p}_1^{\mathcal{S}_0}$. This completes the attack description.

The common setting. Setting $\alpha_i = i$ for $i \in [n]$, taking $i_0 = 2$, we have $\beta_1 = -2(i_1 - 1)$ and $\gamma_1 = -i_1$. Both theses values are small as $n \ll q$, and they are co-prime when i_1 is odd. The attack then proceeds by letting i_1 vary over odd integers in $[n]$. Hence, as soon as n is sufficiently large, this leads to a simple key-recovery attack.

Remarks. A proof of this scheme is sketched in a subsequent work [MCPT24, Section 4.2.5]. It is argued that security would derive from $\mathsf{p}_1^{\mathcal{S}} = \lambda_1^{\mathcal{S}} a \cdot \mathsf{sh}_1 + e$ being indistinguishable from uniform under Ring-LWE. First, this ignores that other decryption shares for the same $\mathcal{S}$ and a may be known to the adversary, giving it a broader view than in the Ring-LWE experiment: this difficulty is typically solved by noise smudging, i.e., choosing a large $\mathsf{PartDec}$ noise e (see [BGG+18]), whereas Ring-LWE remains secure with smaller noise distributions. Second, and more closely related to the attack above, the attacker may obtain $\mathsf{p}_1^{\mathcal{S}}$ for different decryption sets $\mathcal{S}$, providing the attacker with correlated $\mathsf{p}_1^{\mathcal{S}}$'s which cannot be argued to be jointly indistinguishable from uniform under Ring-LWE: indeed, the quantity $\lambda_1^{\mathcal{S}}$ varies whereas a and sh_1 remain the same.

It is suggested in [MBH23, Section 3.4], though not detailed, that the scheme is vulnerable to faulty decryptors, in the sense that revealing two (partial) decryptions of the same ciphertext would reveal some secret information. If one

considers partial decryptions for the same set of decryptors, then noise smudging guarantees that two such partial decryptions do not provide any useful information. However, if one considers partial decryptions for distinct sets of decryptors, then indeed, there are attacks: the one we described above fits this scenario.

This is also the scenario considered in the subsequent work [MCPT24, Section 4.3]. In this work, it is argued that in case of a faulty decryptor, one may retry with a new set of decryptors and an eavesdropper obtains two shares $\mathsf{p}_1^{\mathcal{S}_0} = \lambda_1^{\mathcal{S}_0} a \cdot \mathsf{sh}_1 + e_0$ and $\mathsf{p}_1^{\mathcal{S}_1} = \lambda_1^{\mathcal{S}_1} a \cdot \mathsf{sh}_1 + e_1$. The attacker may then multiply by the inverses of the Lagrange coefficients and distinguish the samples from uniform. Syntactically, this would also qualify as a wk-IND-CPA attack. However, the attack is incomplete, as the inverses modulo q of the Lagrange coefficients are not small, and it is unclear how distinguishing would proceed. As already discussed in Section 1.3, the difficulty is also pointed in [OT25, Section 2.3] and illustrated with a start of an attack, but the latter is also incomplete and we do not see how to complete it.

To avoid decrypting the same ciphertext twice, one could block retries by keeping track of the ciphertexts that have been considered so far. One could also consider randomizing the ciphertext the second time it appears, by adding a fresh encryption of 0 to it before retrying decryption with a new set of decryptors. These approaches are not considered in [MBH23, MCPT24], but their insecurity is illustrative of the difficulties we are facing. We observe that, by using homomorphic addition, such countermeasures can be circumvented. Indeed, an attacker can obtain two partial decryptions of a ciphertext ct for two distinct set of decryptors as follows: first request a decryption of ct for the first set (no randomization is involved as this is the first time ct is decrypted); then, request partial decryptions of $\mathsf{ct} + \mathsf{ct}'$ and ct' for the second set of decryptors, for any ciphertext ct' obtained from querying the encryption (or evaluation) oracle on an input of the attacker's choice; last, compute the difference of the decryption shares of $\mathsf{ct} + \mathsf{ct}'$ and ct'. As PartDec is linear up to noise, this provides a second partial decryption of ct for the second set of decryptors.

4.2 The HELIUM ThFHE Scheme from [MCPT24]

The scheme from [MCPT24] aims to make the one from [MBH23] secure in the context of decryption retrials when a decryption server does not provide the decryption share it was supposed to. To avoid decrypting the same ciphertext twice, it proposes adding a public-key encryption of 0 to the ciphertext before decrypting it. We prove that it is not secure for the weak security model (wk-IND-CPA).

Scheme description. As in the scheme from [MBH23], the secret key is an element $\mathsf{sk} \in \mathcal{R}_q$ that is Shamir-secret-shared as $(\mathsf{sh}_i)_{i\in[n]}$ across the parties. A ciphertext for a plaintext $\mu \in \mathcal{R}$ is still of the form $\mathsf{ct} = (a, b) \in \mathcal{R}_q^2$, with $b + a \cdot \mathsf{sk} = \Delta\mu + e \bmod q$ and $\|e\| \ll q$. To decrypt ct, the parties generate an encryption of 0 using the public key and add it to ct. The public key is of the form $\mathsf{pk} = (a_{\mathsf{pk}}, b_{\mathsf{pk}}) \in \mathcal{R}_q^2$ with $b_{\mathsf{pk}} + a_{\mathsf{pk}} \cdot \mathsf{sk} = e_{\mathsf{pk}} \bmod q$ and $\|e_{\mathsf{pk}}\| \ll q$. An

encryption of 0 is obtained from pk by computing:

$$\mathsf{ct}_0 = (a_0, b_0) := u \cdot \mathsf{pk} + (f, g) = (u \cdot a_{\mathsf{pk}} + f, u \cdot b_{\mathsf{pk}} + g) \ ,$$

where $u, f, g \in \mathcal{R}$ satisfy $\|u\|, \|f\|, \|g\| \ll q$. It may be checked that ct_0 indeed decrypts to 0 and that $\mathsf{ct} + \mathsf{ct}_0$ decrypts to μ. Decryption then proceeds as in the scheme from [MBH23] with $\mathsf{ct} + \mathsf{ct}_0$: for a given target set $\mathcal{S}$, each party $i \in \mathcal{S}$ computes $\mathsf{p}_i^{\mathcal{S}} = \lambda_i^{\mathcal{S}}(a + a_0) \cdot \mathsf{sh}_i + e_i$, and those decryption shares can be added modulo q together with $b + b_0$, and finally rounded to recover μ.

To ensure correctness, the ciphertexts $\mathsf{ct} + \mathsf{ct}_0$ and hence ct_0 must be the same across all parties participating in decryption. It is proposed in [MCPT24] to deterministically derive the encryption randomness u, f and g from a public seed. The precise instantiation is not specified, but one could imagine that one derives bits from applying an extendable-output function (such as SHAKE256) and then applies a deterministic transformation on input $(\mathsf{ct}, \mathcal{S})$ to those bits to obtain properly distributed ring elements u, f and g. For the attack, it is important to note that u, f and g are known to the attacker.

This scheme variant thwarts the attack above since the partial decryptions no longer have the same form, and the ciphertext encrypting 0 that is added also depends on the set of decryptors.

Cryptanalysis. The attack may be viewed as a more complex extension of the one described for the scheme from [MBH23]. We similarly consider a single ciphertext $\mathsf{ct} = (a, b)$ for plaintext $\mu = 0$, on which we make multiple $\mathsf{PartDec}$ queries for the first user and diverse target sets of decryptors to which it belongs. The goal is to recover the partial secret key sh_1 of the first user. If $t - 1$ other users have been corrupted, then sh_1 and the $t-1$ other partial secret keys can be combined to recover sk and hence allow the attacker to decrypt any ciphertext.

The attack is more complex as ct is combined with a different encryption of 0 before running decryption. Concretely, when making a $\mathsf{PartDec}$ query for the first user and a target set $\mathcal{S}$, the attacker obtains:

$$\mathsf{p}_1^{\mathcal{S}} = \lambda_1^{\mathcal{S}}(a + u^{\mathcal{S}} \cdot a_{\mathsf{pk}} + g^{\mathcal{S}})\mathsf{sh}_1 + e^{\mathcal{S}} \bmod q \ ,$$

where $\lambda_1^{\mathcal{S}}$ is the recombination coefficient. In this equation, the quantities $u^{\mathcal{S}}$, $g^{\mathcal{S}}$ and $e^{\mathcal{S}}$ are small and depend on $\mathcal{S}$, $\lambda_1^{\mathcal{S}}$ has no reason to be small and depends on $\mathcal{S}$, and a, a_{pk} and sh_1 have no reason to be small and are independent of $\mathcal{S}$. Importantly, the only quantities unknown to the attacker are sh_1 and $e^{\mathcal{S}}$.

The first step is to (non-trivially) remove the terms that are not small, by using small non-zero combinations of different $\mathsf{p}_1^{\mathcal{S}}$'s for various $\mathcal{S}$'s. Concretely, the attacker makes four queries with distinct target sets $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ and $\mathcal{S}_4$. We can rewrite $\mathsf{p}_1^{\mathcal{S}_1}$, $\mathsf{p}_1^{\mathcal{S}_2}$, $\mathsf{p}_1^{\mathcal{S}_3}$ and $\mathsf{p}_1^{\mathcal{S}_4}$ as follows (over $\mathcal{R}_q$):

$$\left[\mathsf{p}_1^{\mathcal{S}_1}\ \mathsf{p}_1^{\mathcal{S}_2}\ \mathsf{p}_1^{\mathcal{S}_3}\ \mathsf{p}_1^{\mathcal{S}_4}\right] = \mathsf{sh}_1 \cdot \left[a\ a_{\mathsf{pk}}\ 1\right] \cdot \mathbf{M} + \left[e^{\mathcal{S}_1}\ e^{\mathcal{S}_2}\ e^{\mathcal{S}_3}\ e^{\mathcal{S}_4}\right] \ , \tag{2}$$

where

$$\mathbf{M} = \begin{bmatrix} \lambda_1^{\mathcal{S}_1} & \lambda_1^{\mathcal{S}_2} & \lambda_1^{\mathcal{S}_3} & \lambda_1^{\mathcal{S}_4} \\ \lambda_1^{\mathcal{S}_1} u^{\mathcal{S}_1} & \lambda_1^{\mathcal{S}_2} u^{\mathcal{S}_2} & \lambda_1^{\mathcal{S}_3} u^{\mathcal{S}_3} & \lambda_1^{\mathcal{S}_4} u^{\mathcal{S}_4} \\ \lambda_1^{\mathcal{S}_1} g^{\mathcal{S}_1} & \lambda_1^{\mathcal{S}_2} g^{\mathcal{S}_2} & \lambda_1^{\mathcal{S}_3} g^{\mathcal{S}_3} & \lambda_1^{\mathcal{S}_4} g^{\mathcal{S}_4} \end{bmatrix} \ .$$

The next step for the attacker is to find a small non-zero vector $\mathbf{x}$ such that $\mathbf{M} \cdot \mathbf{x} = \mathbf{0}$. Multiplying Equation (2) on the right by $\mathbf{x}$ will then give

$$x_1 \mathsf{p}_1^{\mathcal{S}_1} + x_2 \mathsf{p}_1^{\mathcal{S}_2} + x_3 \mathsf{p}_1^{\mathcal{S}_3} + x_4 \mathsf{p}_1^{\mathcal{S}_4} = x_1 e_1^{\mathcal{S}_1} + x_2 e_1^{\mathcal{S}_2} + x_3 e_1^{\mathcal{S}_3} + x_4 e_1^{\mathcal{S}_4} \ , \quad (3)$$

where the right-hand side is small compared to q and hence known over $\mathcal{R}$ rather than over $\mathcal{R}_q$. We split the rest of the attack into two components: how to find $\mathbf{x}$, and how to recover sh_1 afterwards.

Finding $\mathbf{x}$. As in the attack against the scheme from [MBH23], we use the fact that ratios of reconstruction coefficients can be expressed as ratios of small numerators and denominators. Let us choose $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ and $\mathcal{S}_4$ such that

$$\mathcal{S}_1 = \mathcal{S}^* \cup \{i_1\},\ \mathcal{S}_2 = \mathcal{S}^* \cup \{i_2\},\ \mathcal{S}_3 = \mathcal{S}^* \cup \{i_3\} \ \text{ and } \ \mathcal{S}_4 = \mathcal{S}^* \cup \{i_4\} \ ,$$

for some set $\mathcal{S}^*$ with $|\mathcal{S}^*| = t - 1$ and containing the first user, and where i_1, i_2, i_3 and i_4 are all distinct. Then we can write, for $k \in \{2, 3, 4\}$:

$$\frac{\lambda_1^{\mathcal{S}_1}}{\lambda_1^{\mathcal{S}_k}} = \frac{\beta_k}{\gamma_k} \quad \text{with} \quad \beta_k = \alpha_{i_1}(\alpha_1 - \alpha_{i_k}) \quad \text{and} \quad \gamma_k = \alpha_{i_k}(\alpha_1 - \alpha_{i_1}) \ , \quad (4)$$

Recall that the α_i's are the interpolation points of Shamir secret sharing. Now, observe that:

$$\mathbf{M} \cdot \begin{bmatrix} -\gamma_2 & -\gamma_3 & -\gamma_4 \\ \beta_2 & 0 & 0 \\ 0 & \beta_3 & 0 \\ 0 & 0 & \beta_4 \end{bmatrix} = \lambda_1^{\mathcal{S}_1} \cdot \begin{bmatrix} 0 & 0 & 0 \\ \gamma_2(u^{\mathcal{S}_2} - u^{\mathcal{S}_1}) & \gamma_3(u^{\mathcal{S}_3} - u^{\mathcal{S}_1}) & \gamma_4(u^{\mathcal{S}_4} - u^{\mathcal{S}_1}) \\ \gamma_2(g^{\mathcal{S}_2} - g^{\mathcal{S}_1}) & \gamma_3(g^{\mathcal{S}_3} - g^{\mathcal{S}_1}) & \gamma_4(g^{\mathcal{S}_4} - g^{\mathcal{S}_1}) \end{bmatrix} \quad (5)$$

Now, if the interpolation points α_i are small, so are the γ_k's, and so is the right-most matrix in the equation above. Using cross-products, we can find a small vector $\mathbf{y} = (y_2, y_3, y_4) \in \mathcal{R}^3$ that belongs to its right kernel. Specifically, letting $a_i = \gamma_i(u^{\mathcal{S}_i} - u^{\mathcal{S}_1})$ and $b_i = \gamma_i(g^{\mathcal{S}_i} - g^{\mathcal{S}_1})$ for $i \in \{2, 3, 4\}$ one can set $y_2 = a_3 b_4 - a_4 b_3$, $y_3 = a_4 b_2 - a_2 b_4$, and $y_3 = a_2 b_3 - a_3 b_2$. Overall, we can set:

$$\mathbf{x} = \begin{bmatrix} -\gamma_2 & -\gamma_3 & -\gamma_4 \\ \beta_2 & 0 & 0 \\ 0 & \beta_3 & 0 \\ 0 & 0 & \beta_4 \end{bmatrix} \cdot \mathbf{y} = \begin{bmatrix} -\gamma_2 y_2 - \gamma_3 y_3 - \gamma_4 y_4 \\ \beta_2 y_2 \\ \beta_3 y_3 \\ \beta_4 y_4 \end{bmatrix} \ . \quad (6)$$

Using $\mathbf{x}$. The attacker uses $\mathbf{x}$ as explained in Equation (3), to obtain $x_1 e_1^{\mathcal{S}_1} + x_2 e_1^{\mathcal{S}_2} + x_3 e_1^{\mathcal{S}_3} + x_4 e_1^{\mathcal{S}_4}$ over $\mathcal{R}$. It is not immediate that such a quantity can be helpful, as the $e_1^{\mathcal{S}_k}$'s are not very small (these are smudging noise terms). However, the vector $\mathbf{x}$ is not arbitrary and the attacker has some control over it. In particular, it can choose the i_k's that determine how the $\mathcal{S}_k$'s differ. Assume it chooses them such that $\alpha_1 - \alpha_{i_2}$ divides both $\alpha_1 - \alpha_{i_3}$ and $\alpha_1 - \alpha_{i_4}$. In that case, the quantity β_2 divides both β_3 and β_4, and hence x_2, x_3 and x_4. Reducing $x_1 e_1^{\mathcal{S}_1} + x_2 e_1^{\mathcal{S}_2} + x_3 e_1^{\mathcal{S}_3} + x_4 e_1^{\mathcal{S}_4}$ modulo β_2 gives $x_1 e_1^{\mathcal{S}_1} \bmod \beta_2$.

Examining the form of γ_k reveals that it can be coprime with β_2. Going further, the vector $\mathbf{y}$ is built using a matrix that depends on the γ_k's and quantities unrelated to β_2: it is likely that its components are coprime with β_2. As a result, the first coordinate x_1 is also likely to be coprime with β_2. This ensures that $x_1 e_1^{\mathcal{S}_1} \bmod \beta_2$ reveals $e_1^{\mathcal{S}_1} \bmod \beta_2$.

By choosing other sets $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ and $\mathcal{S}_4$, the attacker can obtain $e_1^{\mathcal{S}_1}$ modulo other β_2's. The Chinese Remainder Theorem then allows the attacker to recover $e_1^{\mathcal{S}_1}$. This gives sh_1, by Equation (2).

The common setting. Let us focus again on the common setting in which $\alpha_i = i$ for $i \in [n]$. Then, smallness of β_k, γ_k for $k \in \{2,3,4\}$ immediately follows by inspection of Equation (4) via smallness of $\alpha_i = i$, for $i \in [n]$. Similarly, smallness of y_k, for $k \in \{2,3,4\}$ follows from Equation (5) and smallness of $u^{\mathcal{S}_j}$ and $g^{\mathcal{S}_j}$ for $j \in [4]$. Finally, smallness of $\mathbf{x}$ follows from Equation (6).

We further need that $1 - i_2$ divides both $1 - i_3$ and $1 - i_4$. Since the attacker can choose freely i_2, i_3, i_4, this is easily enforced. To complete the attack, one just needs x_1 to be co-prime with β_2. As explained above, this (heuristically) happens with high-probability: it is built from components that are all unrelated to β_2.

Remark. An informal security proof for the scheme is provided in [MCPT24, Section 3.4]. The authors invoke [BV11, Lemma 4]. However, this result is not sufficient to prove the security of the protocol: this lemma assumes that the adversary is not aware of the randomness used to re-randomize the ciphertext, while the adversary actually knows it in HELIUM's framework.

5 Main Construction

In this section, we construct a target-decryptors ThFHE scheme that is stg-SIM-secure (Definition 3.3), by combining an FHE scheme with Shamir secret sharing and PRFs. Our technique is inspired from the PRF-based masking used in the threshold signature construction from [dPKM+24].

5.1 Construction

We use a PRF F with keyspace $\mathcal{K}$, domain $\mathbb{S}_{t,n} \times \mathcal{R}_q^2$ and range $\mathcal{R}_q$. We also rely on the Shamir secret sharing algorithm Share (see Section 2.3). The construction is based on an FHE scheme $\mathsf{FHE} = (\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$ which has the following classical structure: a ciphertext encrypting a plaintext μ is a pair $(a, b) \in \mathcal{R}_q^2$ such that, given the secret key $\mathsf{sk} \in \mathcal{R}_q$, decrypting is done by computing $a \cdot \mathsf{sk} + b$ (and possibly applying a post-processing operation such as rounding). Specifically, we assume we have:

$$a \cdot \mathsf{sk} + b = \mathsf{Encode}(\mu) + e \ ,$$

for $e \in \mathcal{R}_q$ with $\|e\|_\infty \leq B_{\mathsf{Dec}}$, and where Encode is an efficient encoding function associated with an efficient inverse function Decode.

We define $\mathcal{D}_{\mathsf{smdg}} = U([-B_{SM}, B_{SM}])$ where $B_{SM} = \Omega(2^\lambda \cdot B_{\mathsf{Dec}})$, and assume that FHE is compatible with $(n, \mathcal{D}_{\mathsf{smdg}})$, in the sense that for a ciphertext (a, b) as above, the ciphertext $(a, b+e^{\mathsf{smdg}})$ still decrypts to the same message μ with overwhelming probability over $e^{\mathsf{smdg}} = \sum_{1 \le i \le n} e_i^{\mathsf{smdg}}$ where $e_i^{\mathsf{smdg}} \leftarrow \mathcal{D}_{\mathsf{smdg}}$ for all i.

Our target-decryptors ThFHE construction $\mathsf{td\text{-}ThFHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$ is described in Figure 4. KeyGen runs $\mathsf{FHE.KeyGen}$, secret-shares the output secret key and generates n^2 PRF keys $(k_{i,j})_{i,j\in[n]}$. The public key is the FHE public key, and the i-th user's partial decryption key consists of the corresponding share of the FHE secret key and the $2n$ keys $(k_{i,j})_{j\in[n]}$ and $(k_{j,i})_{j\in[n]}$ (note that $k_{i,i}$ is repeated and could actually be ignored; we keep it to slightly simplify the scheme description). Encryption and evaluation are as in the FHE scheme. Decryption for target decryptors $\mathcal{S}$ and user $i \in \mathcal{S}$ considers the term $\lambda_i^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}}$ as in the scheme from [MBH23], but adds a mask $r_i^{\mathcal{S},\mathsf{ct}}$ to it. The mask $r_i^{\mathcal{S},\mathsf{ct}}$ is obtained by subtracting the sum of the PRF evaluations in $(\mathcal{S}, \mathsf{ct})$ for keys $(k_{j,i})_{j\in\mathcal{S}}$ from the sum of the PRF evaluations for keys $(k_{i,j})_{j\in\mathcal{S}}$. As in [MBH23], decryption consists in summing the decryption shares and adding the second ciphertext component.

Overhead. Overall, our construction introduces a small overhead per party compared to the prior (insecure) constructions: each party additionally stores $2(n-1)$ PRF keys and performs $2(n-1)$ PRF evaluations during $\mathsf{PartDec}$. Relying on AES yields a $256 \cdot (n-1)$-bit overhead for the secret key. Note that typical FHE parameters set $N = 2^{16}$ and $\log q \ge 2^7$ (to support noise flooding), so P_i's share sh_i of the FHE secret key sk still dominates sk_i up to $n = 2^{16} = 65536$ parties. This ignores the much larger public and evaluation keys. Moreover, generating the additional masks during $\mathsf{PartDec}$ takes about $2n$ milliseconds on single-thread CPU, based on conservative estimates for AES throughput (1MB/ms).

The rest of the section is devoted to proving the following result.

Theorem 5.1. *Assume that F is a secure PRF and that* FHE *is correct, compact and* $\mathsf{IND\text{-}CPA}$*-secure. Assume further that $(n, \mathcal{D}_{\mathsf{smdg}})$ is compatible with* FHE*. Then* $\mathsf{td\text{-}ThFHE}$ *is correct, compact and* $\mathsf{stg\text{-}SIM}$*-secure.*

5.2 Correctness and Compactness

Correctness. Let $\mathsf{ct} = (a, b)$ be a ciphertext, so that we have $b = -a \cdot \mathsf{sk} + \mu + e$ for some error e with $\|e\|_\infty \le B_{\mathsf{Dec}}$. Let $\mathcal{S} \in \mathbb{S}_{t,n}$ and $\mathsf{p}_i^{\mathcal{S}} = \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$, for $i \in \mathcal{S}$. We have:

$$\mathsf{p}_i^{\mathcal{S}} = \lambda_i^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}} + r_i^{\mathcal{S},\mathsf{ct}} \ ,$$

$\mathsf{KeyGen}(1^\lambda, 1^n, 1^t)$:

1 $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$
2 **for** $i, j \in [n]$ **do**
3 $\quad k_{i,j} \leftarrow U(\mathcal{K})$
4 $(\mathsf{sh}_1, \ldots, \mathsf{sh}_n) \leftarrow \mathsf{Share}(n, t, \mathsf{sk})$
5 **for** $1 \leq i \leq n$ **do**
6 $\quad \mathsf{sk}_i \leftarrow (i, \mathsf{sh}_i, (k_{i,j})_{j\in[n]}, (k_{j,i})_{j\in[n]})$
7 **return** $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{pk})$

$\mathsf{Enc}(\mathsf{pk}, \mu)$:

1 **return** $\mathsf{FHE.Enc}(\mathsf{pk}, \mu)$

$\mathsf{Eval}(\mathsf{pk}, \mathsf{C}, (\mathsf{ct}_j)_{j\in[\ell]})$:

1 **return** $\mathsf{FHE.Eval}(\mathsf{pk}, \mathsf{C}, (\mathsf{ct}_j)_{j\in[\ell]})$

$\mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$:

1 Parse ct as (a, b)
2 Parse sk_i as $(i, \mathsf{sh}_i, (k_{i,j})_{j\in[n]}, (k_{j,i})_{j\in[n]})$
3 **if** $i \notin \mathcal{S}$ **then**
4 $\quad$ **return** $\perp$
5 $e_i^{\mathsf{smdg}} \leftarrow \mathcal{D}_{\mathsf{smdg}}$
6 **for** $j \in \mathcal{S}$ **do**
7 $\quad r_{i,j}^{\mathcal{S},\mathsf{ct}} \leftarrow F_{k_{i,j}}(\mathcal{S}, \mathsf{ct})$
8 $\quad r_{j,i}^{\mathcal{S},\mathsf{ct}} \leftarrow F_{k_{j,i}}(\mathcal{S}, \mathsf{ct})$
9 $r_i^{\mathcal{S},\mathsf{ct}} \leftarrow \sum_{j\in\mathcal{S}}(r_{i,j}^{\mathcal{S},\mathsf{ct}} - r_{j,i}^{\mathcal{S},\mathsf{ct}})$
10 $\mathsf{p}_i^{\mathcal{S}} \leftarrow \lambda_i^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}} + r_i^{\mathcal{S},\mathsf{ct}}$
11 **return** $\mathsf{p}_i^{\mathcal{S}}$

$\mathsf{FinDec}(\mathsf{ct}, (\mathsf{p}_i^{\mathcal{S}})_{i\in\mathcal{S}})$:

1 Parse ct as (a, b)
2 **return** $b + \sum_{i\in\mathcal{S}} \mathsf{p}_i^{\mathcal{S}}$

Fig. 4. The td-ThFHE scheme.

where $r_i^{\mathcal{S},\mathsf{ct}} = \sum_{j\in\mathcal{S}}(r_{i,j}^{\mathcal{S},\mathsf{ct}} - r_{j,i}^{\mathcal{S},\mathsf{ct}})$. Then the following holds:

$$\begin{aligned}\mathsf{FinDec}(\mathsf{ct}, (\mathsf{p}_i^{\mathcal{S}})_{i\in\mathcal{S}}) &= b + \sum_{i\in\mathcal{S}}(\lambda_i^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}} + r_i^{\mathcal{S},\mathsf{ct}}) \\ &= b + a \cdot \sum_{i\in\mathcal{S}}(\lambda_i^{\mathcal{S}} \cdot \mathsf{sh}_i) + \sum_{i\in\mathcal{S}} e_i^{\mathsf{smdg}} + \sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}(r_{i,j}^{\mathcal{S},\mathsf{ct}} - r_{j,i}^{\mathcal{S},\mathsf{ct}}) \ .\end{aligned}$$

By the properties of Shamir secret sharing, the first sum is exactly sk. Also, the last sum is equal to 0. This leads us to:

$$\mathsf{FinDec}(\mathsf{ct}, (\mathsf{p}_i^{\mathcal{S}})_{i\in\mathcal{S}}) = b + a \cdot \mathsf{sk} + \sum_{i\in\mathcal{S}} e_i^{\mathsf{smdg}} = \mu + \left(e + \sum_{i\in\mathcal{S}} e_i^{\mathsf{smdg}}\right) \ .$$

Correctness follows from that of FHE and the assumption that $(n, \mathcal{D}_{\mathsf{smdg}})$ is compatible with FHE.

Compactness. Compactness follows from that of the underlying FHE scheme, as we do not alter the encryption and evaluation algorithms.

5.3 Security

IND-CPA security follows from that of the underlying FHE scheme since we do not alter the encryption and evaluation algorithms. For many existing FHE schemes, security is based on the Ring-LWE (or LWE) assumption (and circular-security due to the evaluation key).

Before moving to the simulation-based security proof, we fix some notation. To improve readability, we sometimes refer to a partial decryption query $(j, i, \mathcal{S})$ where ct is the ciphertext stored in $\mathsf{L}[j]$ as a partial decryption query $(\mathsf{ct}, i, \mathcal{S})$ or a partial decryption query of ciphertext ct for $(i, \mathcal{S})$.

Further, recall that a partial decryption share $\mathsf{p}_i^{\mathcal{S}}$ has the form:

$$\mathsf{p}_i^{\mathcal{S}} = \lambda_i^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}} + r_i^{\mathcal{S},\mathsf{ct}} \ ,$$

where $r_i^{\mathcal{S},\mathsf{ct}} = \sum_{j \in \mathcal{S}} (F_{k_{i,j}}(\mathcal{S}, \mathsf{ct}) - F_{k_{j,i}}(\mathcal{S}, \mathsf{ct}))$. It is therefore formed of the reconstruction term of Shamir secret sharing $\lambda_i^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_i$, a smudging error e_i^{smdg}, and the pseudorandom mask $r_i^{\mathcal{S},\mathsf{ct}}$. We define $\mathsf{q}_i^{\mathcal{S}}$ as the (deterministic) form of this partial decryption share without smudging noise, i.e.:

$$\mathsf{q}_i^{\mathcal{S}} := \lambda_i^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_i + r_i^{\mathcal{S},\mathsf{ct}} \ . \tag{7}$$

The latter term can be computed by the adversary if it has corrupted sk_i.

We turn to the simulation property of $\mathsf{stg\text{-}SIM}$-security. We define our simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1, \mathsf{Sim}_2)$ in Figure 5. The purpose of Sim_0 is to distribute well-generated key shares, given the public key pk, which is achieved without using pk. Concerning Sim_1, recall that it is used when the adversary does not have enough partial decryption shares (by corruption or by queries) to recover the plaintext. For this reason, Sim_1 outputs a uniformly distributed decryption share, without even using ct. It does so in a way that may seem contrived at first: the smudging error e_i^{smdg} is added to a term that is independent and uniform, and is hence useless to get a uniform output. This is because the adversary can make multiple queries for the same input $(\mathsf{ct}, i, \mathcal{S})$: in this case, the term $\mathsf{q}_i^{\mathcal{S}}$ is common to all replies and their difference is still non-zero because of the smudging errors. Finally, Sim_2 is used when the adversary has enough corrupted users and partial decryption shares so that this query will allow it to recover the plaintext, i.e., when $\mathsf{CanCombine}(\mathcal{S}_{\mathsf{cor}}, \mathsf{P}[\mathsf{ct}], i, \mathcal{S}) = 1$. We emphasize that, by convention, this can only occur if $i \notin \mathcal{S}_{\mathsf{cor}}$. Otherwise, the partial decryption can be generated using the corrupted secret key sk_i. Sim_2 computes the $\mathsf{q}_\ell^{\mathcal{S}}$'s that the adversary already knows, i.e., those corresponding to corrupted ℓ's, then it retrieves the $\mathsf{q}_\ell^{\mathcal{S}}$'s corresponding to prior decryption queries, and finally it returns a decryption share for user i that is consistent with plaintext μ.

We now show that $\mathsf{Exp}_{\mathsf{real}}^{\mathsf{stg\text{-}SIM}}$ and $\mathsf{Exp}_{\mathsf{ideal}}^{\mathsf{stg\text{-}SIM}}$ are computationally indistinguishable. We proceed via a sequence of hybrid experiments. In each hybrid, we gently modify how the oracle queries are answered by the challenger. The main changes are made within the decryption oracle, and we describe the decryption oracles of these hybrids in Figure 6. In all hybrid experiments, we assume st is initialized during KeyGen as the set of all corrupted partial decryption keys.

$\mathbf{Hyb}_0$. This is the real experiment $\mathsf{Exp}_{\mathsf{real}}^{\mathsf{stg\text{-}SIM}}$, described in Figure 3. In this experiment, partial decryption queries are answered via the ODec oracle, which generates partial decryption shares by honestly using $\mathsf{PartDec}$. This corresponds to ODec_0 in Figure 6.

$\mathsf{Sim}_0(\mathsf{pk}, \mathcal{S}_{\mathsf{cor}})$:

$(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_n) \leftarrow \mathsf{Share}(n, t, 0)$
for $i \in \mathcal{S}_{\mathsf{cor}}$ **do**
  for $j \in [n]$ **do**
    $(k_{i,j}, k_{j,i}) \leftarrow U(\mathcal{K} \times \mathcal{K})$
for $i \in \mathcal{S}_{\mathsf{cor}}$ **do**
  $\mathsf{sk}'_i \leftarrow (i, \mathsf{sh}'_i, (k_{i,j}, k_{j,i})_{j \in [n]})$
$\mathsf{st} \leftarrow \{(\mathsf{pk}, (\mathsf{sk}'_i)_{i \in \mathcal{S}_{\mathsf{cor}}})\}$
return $(\mathsf{st}, (\mathsf{sk}'_i)_{i \in \mathcal{S}_{\mathsf{cor}}})$

$\mathsf{Sim}_1(\mathsf{st}, \mathsf{ct}, i, \mathcal{S})$:

Find $(\mathsf{pk}, (\mathsf{sk}'_i)_{i \in \mathcal{S}_{\mathsf{cor}}})$ in st
if $i \in \mathcal{S}_{\mathsf{cor}}$ **then**
  return $\mathsf{PartDec}(\mathsf{sk}'_i, \mathsf{ct}, \mathcal{S})$
if $\nexists (\mathsf{ct}, i, \mathcal{S}, \mathsf{q}_i^{\mathcal{S}}) \in \mathsf{st}$ **then**
  $\mathsf{q}_i^{\mathcal{S}} \leftarrow U(\mathcal{R}_q)$
  $\mathsf{st} \leftarrow \mathsf{st} \cup \{(\mathsf{ct}, i, \mathcal{S}, \mathsf{q}_i^{\mathcal{S}})\}$
else
  Retrieve $\mathsf{q}_i^{\mathcal{S}}$ from st
$e_i^{\mathsf{smdg}} \leftarrow \mathcal{D}_{\mathsf{smdg}}$
$\mathsf{p}_i^{\mathcal{S}} \leftarrow \mathsf{q}_i^{\mathcal{S}} + e_i^{\mathsf{smdg}}$
return $(\mathsf{st}, \mathsf{p}_i^{\mathcal{S}})$

$\mathsf{Sim}_2(\mathsf{st}, \mathsf{ct}, i, \mathcal{S}, \mu)$:

Find $(\mathsf{pk}, (\mathsf{sk}'_i)_{i \in \mathcal{S}_{\mathsf{cor}}})$ in st
if $i \in \mathcal{S}_{\mathsf{cor}}$ **then**
  return $\mathsf{PartDec}(\mathsf{sk}'_i, \mathsf{ct}, \mathcal{S})$
Parse ct as (a, b)
for $\ell \in \mathcal{S}_{\mathsf{cor}} \cap \mathcal{S}$ **do**
  Parse sk'_ℓ as $(\ell, \mathsf{sh}_\ell, (k_{\ell,j}, k_{j,\ell})_{j \in [n]})$
  $r_\ell^{\mathcal{S},\mathsf{ct}} \leftarrow \sum_{j \in \mathcal{S}} (F_{k_{\ell,j}}(\mathcal{S}, \mathsf{ct}) - F_{k_{j,\ell}}(\mathcal{S}, \mathsf{ct}))$
  $\mathsf{q}_\ell^{\mathcal{S}} \leftarrow \lambda_\ell^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_\ell + r_\ell^{\mathcal{S},\mathsf{ct}}$
for $\ell \in \mathcal{S} \setminus (\mathcal{S}_{\mathsf{cor}} \cup \{i\})$ **do**
  Find $(\mathsf{ct}, \ell, \mathcal{S}, \mathsf{q}_\ell^{\mathcal{S}})$ in st
for $\ell \in \mathcal{S}$ **do**
  $e_\ell^{\mathsf{smdg}} \leftarrow \mathcal{D}_{\mathsf{smdg}}$
$\mathsf{p}_i^{\mathcal{S}} \leftarrow -b + \mu + e_i^{\mathsf{smdg}} - \sum_{\ell \in \mathcal{S} \setminus \{i\}} \mathsf{q}_\ell^{\mathcal{S}}$
return $(\mathsf{st}, \mathsf{p}_i^{\mathcal{S}})$

Fig. 5. Simulator $(\mathsf{Sim}_0, \mathsf{Sim}_1, \mathsf{Sim}_2)$ for the td-ThFHE security proof.

Hyb$_1$. In this hybrid, we modify the decryption oracle and replace it with ODec_1, described in Figure 6. The challenger now keeps the record P updated with the list of partial decryption queries made by the attacker. It also runs $\mathsf{CanCombine}$ based on this record to check whether the attacker can recover the underlying plaintext or not after the current query. These changes are purely internal to ODec_1 and do not affect its responses, which are still computed by running $\mathsf{PartDec}$. Therefore, **Hyb**$_0$ and **Hyb**$_1$ are perfectly indistinguishable.

Before moving to the next hybrid, note that if $\mathsf{CanCombine}(\mathcal{S}_{\mathsf{cor}}, \mathsf{P}[\mathsf{ct}], i, \mathcal{S})$ is 1, then the adversary already has knowledge of partial decryption shares of ct for all $\ell \in \mathcal{S} \setminus (\mathcal{S}_{\mathsf{cor}} \cup \{i\})$ and target decryptors $\mathcal{S}$. It can also compute the other partial decryptions using corrupted partial decryption keys sk_ℓ's for $\ell \in \mathcal{S}_{\mathsf{cor}} \cap \mathcal{S}$ by computing $\mathsf{q}_\ell^{\mathcal{S}} = \lambda_\ell^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_\ell + r_i^{\mathcal{S},\mathsf{ct}}$ as in Equation (7).

$\mathsf{ODec}_0(j, i, \mathcal{S})$:

if $j > \mathsf{ctr} \vee |\mathcal{S}| \neq t \vee i \notin \mathcal{S}$ **then**
  return $\perp$
$(\mu, \mathsf{ct}) \leftarrow \mathsf{L}[j]$
$\mathsf{p}_i^{\mathcal{S}} \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$
return $\mathsf{p}_i^{\mathcal{S}}$

$\mathsf{ODec}_1(j, i, \mathcal{S})$:

if $j > \mathsf{ctr} \vee |\mathcal{S}| \neq t \vee i \notin \mathcal{S}$ **then**
  return $\perp$
$(\mu, \mathsf{ct}) \leftarrow \mathsf{L}[j]$
if $\mathsf{CanCombine}(\mathcal{S}_{\mathsf{cor}}, \mathsf{P}[\mathsf{ct}], i, \mathcal{S})$ **then**
  $\mathsf{p}_i^{\mathcal{S}} \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$
else
  $\mathsf{p}_i^{\mathcal{S}} \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$
$\mathsf{P}[\mathsf{ct}] \leftarrow \mathsf{P}[\mathsf{ct}] \cup \{(i, \mathcal{S})\}$
return $\mathsf{p}_i^{\mathcal{S}}$

$\mathsf{ODec}_2(j, i, \mathcal{S})$:

if $j > \mathsf{ctr} \vee |\mathcal{S}| \neq t \vee i \notin \mathcal{S}$ **then**
  return $\perp$
$(\mu, \mathsf{ct}) \leftarrow \mathsf{L}[j]$
if $\mathsf{CanCombine}(\mathcal{S}_{\mathsf{cor}}, \mathsf{P}[\mathsf{ct}], i, \mathcal{S})$ **then**
  $\mathsf{p}_i^{\mathcal{S}} = -b + \mu + e_i^{\mathsf{smdg}} - \sum_{\ell \in \mathcal{S} \setminus \{i\}} \mathsf{q}_\ell^{\mathcal{S}}$
else
  $\mathsf{p}_i^{\mathcal{S}} \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$
$\mathsf{P}[\mathsf{ct}] \leftarrow \mathsf{P}[\mathsf{ct}] \cup \{(i, \mathcal{S}, \mathsf{p}_i^{\mathcal{S}})\}$
return $\mathsf{p}_i^{\mathcal{S}}$

$\mathsf{ODec}_3(j, i, \mathcal{S})$:

if $j > \mathsf{ctr} \vee |\mathcal{S}| \neq t \vee i \notin \mathcal{S}$ **then**
  return $\perp$
$(\mu, \mathsf{ct}) \leftarrow \mathsf{L}[j]$
if $\mathsf{CanCombine}(\mathcal{S}_{\mathsf{cor}}, \mathsf{P}[\mathsf{ct}], i, \mathcal{S})$ **then**
  $\mathsf{p}_i^{\mathcal{S}} = -b + \mu + e_i^{\mathsf{smdg}} - \sum_{\ell \in \mathcal{S} \setminus \{i\}} \mathsf{q}_\ell^{\mathcal{S}}$
else
  if $i \in \mathcal{S}_{\mathsf{cor}}$ **then**
    return $\mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$
  if $\nexists (\mathsf{ct}, i, \mathcal{S}, \mathsf{q}_i^{\mathcal{S}}) \in \mathsf{st}$ **then**
    $\mathsf{q}_i^{\mathcal{S}} \leftarrow U(\mathcal{R}_q)$
    $\mathsf{st} \leftarrow \mathsf{st} \cup \{(\mathsf{ct}, i, \mathcal{S}, \mathsf{q}_i^{\mathcal{S}})\}$
  else
    Retrieve $\mathsf{q}_i^{\mathcal{S}}$ from st
  $e_i^{\mathsf{smdg}} \leftarrow \mathcal{D}_{\mathsf{smdg}}$
  $\mathsf{p}_i^{\mathcal{S}} \leftarrow \mathsf{q}_i^{\mathcal{S}} + e_i^{\mathsf{smdg}}$
$\mathsf{P}[\mathsf{ct}] \leftarrow \mathsf{P}[\mathsf{ct}] \cup \{(i, \mathcal{S}, \mathsf{p}_i^{\mathcal{S}})\}$
return $\mathsf{p}_i^{\mathcal{S}}$

$\mathsf{ODec}_4(j, i, \mathcal{S})$:

if $j > \mathsf{ctr} \vee |\mathcal{S}| \neq t \vee i \notin \mathcal{S}$ **then**
  return $\perp$
$(\mu, \mathsf{ct}) \leftarrow \mathsf{L}[j]$
if $\mathsf{CanCombine}(\mathcal{S}_{\mathsf{cor}}, \mathsf{P}[\mathsf{ct}], i, \mathcal{S})$ **then**
  $(\mathsf{st}, \mathsf{p}_i^{\mathcal{S}}) \leftarrow \mathsf{Sim}_2(\mathsf{st}, \mathsf{ct}, i, \mathcal{S}, \mu)$
else
  $(\mathsf{st}, \mathsf{p}_i^{\mathcal{S}}) \leftarrow \mathsf{Sim}_1(\mathsf{st}, \mathsf{ct}, i, \mathcal{S})$
$\mathsf{P}[\mathsf{ct}] \leftarrow \mathsf{P}[\mathsf{ct}] \cup \{(i, \mathcal{S}, \mathsf{p}_i^{\mathcal{S}})\}$
return $\mathsf{p}_i^{\mathcal{S}}$

Fig. 6. Partial decryption oracles for the hybrid experiments.

Recall that $\mathsf{ct} = (a, b)$ is such that $a \cdot \mathsf{sk} + b = \mu + e$, so we have:

$$a \cdot \mathsf{sk} = -b + \mu + e \ ,$$

for some small error e with $\|e\|_\infty \leq B_{\mathsf{Dec}}$. Recall also that the queried partial decryption $\mathsf{p}_i^{\mathcal{S}}$ is of the form:

$$\mathsf{p}_i^{\mathcal{S}} \leftarrow \lambda_i^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}} + r_i^{\mathcal{S},\mathsf{ct}} \ .$$

By construction, we have $a \cdot \mathsf{sk} = \sum_{\ell \in \mathcal{S}} \lambda_\ell^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_\ell$ and $\sum_{\ell \in \mathcal{S}} r_\ell^{\mathcal{S},\mathsf{ct}} = 0$. The two equations above then lead to:

$$\begin{aligned}
\mathsf{p}_i^{\mathcal{S}} &= a \cdot \mathsf{sk} - \sum_{\ell \in \mathcal{S} \setminus \{i\}} \lambda_\ell^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_\ell + e_i^{\mathsf{smdg}} - \sum_{\ell \in \mathcal{S} \setminus \{i\}} r_\ell^{\mathcal{S},\mathsf{ct}} \\
&= -b + \mu + e + e_i^{\mathsf{smdg}} - \sum_{\ell \in \mathcal{S} \setminus \{i\}} (\lambda_\ell^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_\ell + r_\ell^{\mathcal{S},\mathsf{ct}}) \\
&= -b + \mu + e + e_i^{\mathsf{smdg}} - \sum_{\ell \in \mathcal{S} \setminus \{i\}} \mathsf{q}_\ell^{\mathcal{S}} \ . \qquad (8)
\end{aligned}$$

Hyb$_2$. In this experiment, we change the decryption oracle from ODec_1 to ODec_2. The only difference is how partial decryptions are computed when the condition $\mathsf{CanCombine}(\mathcal{S}_{\mathsf{cor}}, \mathsf{P}[\mathsf{ct}], i, \mathcal{S}) = 1$ holds. Instead of revealing the appropriate value for $\mathsf{p}_i^{\mathcal{S}}$ as in **Hyb**$_1$ (Equation 8), the challenger computes $\mathsf{p}_i^{\mathcal{S}}$ as:

$$\mathsf{p}_i^{\mathcal{S}} = -b + \mu + e_i^{\mathsf{smdg}} - \sum_{\ell \in \mathcal{S} \setminus \{i\}} \mathsf{q}_\ell^{\mathcal{S}} \ . \qquad (9)$$

Observe that Equations (8) and (9) differ exactly by the evaluation noise e. By Lemma 2.1, we have $\mathrm{SD}(e + e_i^{\mathsf{smdg}}, e_i^{\mathsf{smdg}}) = \mathsf{negl}(\lambda)$, hence the two distributions are statistically close and **Hyb**$_1$ and **Hyb**$_2$ are statistically indistinguishable.

We note that Equation (9) corresponds to how $\mathsf{p}_i^{\mathcal{S}}$ is sampled by Sim_2 at Step 13. The only difference with Sim_2 is that, for now, the $\mathsf{q}_\ell^{\mathcal{S}}$'s for $\ell \notin \mathcal{S}_{\mathsf{cor}}$ are not defined the same way in **Hyb**$_2$ as in Sim_2: the latter uses the $\mathsf{q}_\ell^{\mathcal{S}}$ values computed by Sim_1, whereas the former uses Equation (7).

Let us further analyze the partial decryption shares in **Hyb**$_2$ before moving to **Hyb**$_3$. Let us now focus on partial decryption shares when we have $\mathsf{CanCombine}(\mathcal{S}_{\mathsf{cor}}, \mathsf{P}[\mathsf{ct}], i, \mathcal{S}) = 0$ and $i \notin \mathcal{S}_{\mathsf{cor}}$, i.e., when the adversary cannot produce itself the partial decryption but also cannot decrypt yet, even after receiving the result to the query $(\mathsf{ct}, i, \mathcal{S})$ it is making. For such a query, we have:

$$\mathsf{p}_i^{\mathcal{S}} := \lambda_i^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}} + r_i^{\mathcal{S},\mathsf{ct}} \ ,$$

where $r_i^{\mathcal{S},\mathsf{ct}} := \sum_{j \in \mathcal{S}} (F_{k_{i,j}}(\mathcal{S}, \mathsf{ct}) - F_{k_{j,i}}(\mathcal{S}, \mathsf{ct}))$.

Since $\mathsf{CanCombine}(\mathcal{S}_{\mathsf{cor}}, \mathsf{P}[\mathsf{ct}], i, \mathcal{S}) = 0$, by definition of $\mathsf{CanCombine}$, there exists $\ell \in \mathcal{S}$ such that the following three statements simultaneously hold:

- $\ell \neq i$;
- $\ell \notin \mathcal{S}_{\mathsf{cor}}$;
- no partial decryption query $(\mathsf{ct}, \ell, \mathcal{S})$ has been made before.

For such an index ℓ, note that only sk_i and sk_ℓ use the PRF keys $k_{i,\ell}$ and $k_{\ell,i}$. As $i \notin \mathcal{S}_{\mathsf{cor}}$, these keys are not corrupted. Also, observe that there are only two ways for the adversary to get a partial decryption share that depends on $r_{i,\ell}^{\mathcal{S},\mathsf{ct}} = F_{k_{i,\ell}}(\mathcal{S}, \mathsf{ct})$ and $r_{\ell,i}^{\mathcal{S},\mathsf{ct}} = F_{k_{\ell,i}}(\mathcal{S}, \mathsf{ct})$:

- either it queries $(\mathsf{ct}, i, \mathcal{S})$ to get $\mathsf{p}_i^{\mathcal{S}}$, which contains the term $r_{i,\ell}^{\mathcal{S},\mathsf{ct}} - r_{\ell,i}^{\mathcal{S},\mathsf{ct}}$;

- or it queries $(\mathsf{ct}, \ell, \mathcal{S})$ to get $\mathsf{p}_\ell^{\mathcal{S}}$, which contains the opposite term $r_{\ell,i}^{\mathcal{S},\mathsf{ct}} - r_{i,\ell}^{\mathcal{S},\mathsf{ct}}$.

This observation implies that, for a ciphertext ct and target decryptors $\mathcal{S}$, as long as there is an $\ell \in \mathcal{S} \setminus \mathcal{S}_{\mathsf{cor}}$ for which the adversary has not queried a partial decryption, then every other partial decryption share $(\mathsf{ct}, j, \mathcal{S})$ for $j \in \mathcal{S} \setminus (\mathcal{S}_{\mathsf{cor}} \cup \{\ell\})$ contains a pseudorandom term $r_{j,\ell}^{\mathcal{S},\mathsf{ct}} - r_{\ell,j}^{\mathcal{S},\mathsf{ct}}$ about which the adversary has no information. In particular, this implies that all such partial decryption shares are pseudorandom in the view of the adversary.

$\mathbf{Hyb}_3$. In this experiment, we replace the PRF evaluations for uncorrupted PRF keys $(k_{i,j}, k_{j,i})$ for $i, j \notin \mathcal{S}_{\mathsf{cor}}$ by truly uniform values. This is done by having the challenger lazily sampling all $r_{i,j}$ and $r_{j,i}$ as uniform and independent values and by having the challenger keep the record of all generated values.

Due to the observation above, all partial decryption queries $\mathsf{p}_i^{\mathcal{S}}$ for which $\mathsf{CanCombine}(\mathcal{S}_{\mathsf{cor}}, \mathsf{P}[\mathsf{ct}], i, \mathcal{S}) = 0$ (and $i \notin \mathcal{S}_{\mathsf{cor}}$) are now uniform and independent, since the term $r_{i,\ell}^{\mathcal{S},\mathsf{ct}} - r_{\ell,i}^{\mathcal{S},\mathsf{ct}}$ is now uniform.

To take this change into account, we modify the decryption oracle in this experiment to ODec_3, described in Figure 6. In ODec_3, on query $(\mathsf{ct}, i, \mathcal{S})$ such that $i \notin \mathcal{S}_{\mathsf{cor}}$ and $\mathsf{CanCombine}(\mathcal{S}_{\mathsf{cor}}, \mathsf{P}[\mathsf{ct}], i, \mathcal{S}) = 0$, the challenger now sets the deterministic form of partial decryptions $\mathsf{q}_i^{\mathcal{S}}$ as a uniform value the first time it answers this query, and samples $\mathsf{p}_i^{\mathcal{S}}$ by adding a fresh smudging error $e_i^{\mathsf{smdg}} \leftarrow \mathcal{D}_{\mathsf{smdg}}$ to $\mathsf{q}_i^{\mathcal{S}}$. The challenger keeps track of previously defined $\mathsf{q}_i^{\mathcal{S}}$ values by adding $(\mathsf{ct}, i, \mathcal{S}, \mathsf{q}_i^{\mathcal{S}})$ to the state st everytime it fixes a value $\mathsf{q}_i^{\mathcal{S}}$. $\mathbf{Hyb}_2$ and $\mathbf{Hyb}_3$ are computationally indistinguishable based on the pseudorandomness of F.

$\mathbf{Hyb}_4$. This is the hybrid in which the partial decryption oracle is changed to ODec_4, which precisely corresponds to oracle OSim from $\mathsf{Exp}_{\mathsf{ideal}}^{\mathsf{stg\text{-}SIM}}$. This is actually just a syntactical change as ODec_3 is already implicitly running Sim_1 and Sim_2. Therefore, $\mathbf{Hyb}_3$ and $\mathbf{Hyb}_4$ are perfectly indistinguishable.

$\mathbf{Hyb}_5$. In this hybrid, we change the generation of the corrupted partial decryption keys to $\mathsf{Sim}_0(\mathsf{pk}, \mathcal{S}_{\mathsf{cor}})$: the challenger samples the corrupted shares $(\mathsf{sh}_i)_{i \in \mathcal{S}_{\mathsf{cor}}}$ as shares of 0 rather than as shares of sk. Note that it relies on OSim to answer all decryption queries as in $\mathbf{Hyb}_4$, and thus does not rely on the knowledge of any information about sk beyond the corrupted partial decryption keys $(\mathsf{sk}_i)_{i \in \mathcal{S}_{\mathsf{cor}}}$. $\mathbf{Hyb}_4$ and $\mathbf{Hyb}_5$ are indistinguishable by the privacy property of Shamir secret sharing (see Section 2.3). $\mathbf{Hyb}_5$ is actually the ideal experiment $\mathsf{Exp}_{\mathsf{ideal}}^{\mathsf{stg\text{-}SIM}}$ described in Figure 3, with the simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1, \mathsf{Sim}_2)$ from Figure 5.

By a hybrid argument, the experiments $\mathsf{Exp}_{\mathsf{real}}^{\mathsf{stg\text{-}SIM}}$ and $\mathsf{Exp}_{\mathsf{ideal}}^{\mathsf{stg\text{-}SIM}}$ are computationally indistinguishable and $\mathsf{td\text{-}ThFHE}$ is $\mathsf{stg\text{-}SIM}$-secure. This completes the proof of Theorem 5.1. □

Acknowledgements. We thank Dan Boneh and Changmin Lee for helpful discussions. This work was undertaken while the first author was an intern at CryptoLab Inc.

References

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.

BBN+25. Dan Boneh, Benedikt Bünz, Kartik Nayak, Lior Rotem, and Victor Shoup. Context-dependent threshold decryption and its applications. Cryptology ePrint Archive, Paper 2025/279, 2025.

BFM+25. Zvika Brakerski, Offir Friedman, Avichai Marmor, Dolev Mutzari, Yuval Spiizer, and Ni Trieu. Threshold FHE with efficient asynchronous decryption. *Cryptology ePrint Archive*, 2025.

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 *CRYPTO*, 2018.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.

BLMR15. Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. 2015.

Bra12. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO*, 2012.

BV11. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, 2011.

CGGI16. Ilaria Chillotti, Nicolas Gama, Maryia Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *ASIACRYPT*, 2016.

CKKS17. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT*, 2017.

CKL24. Wonhee Cho, Jiseung Kim, and Changmin Lee. (In)Security of threshold fully homomorphic encryption based on Shamir secret sharing. Cryptology ePrint Archive, Paper 2024/1858, 2024.

dPKM+24. Rafael del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani Saarinen. Threshold Raccoon: Practical threshold signatures from standard lattice assumptions. In *EUROCRYPT*, 2024.

EY24. Ehsan Ebrahimi and Anshu Yadav. Strongly secure universal thresholdizer. In *ASIACRYPT*, 2024.

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

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009.

GKS24. Kamil Doruk Gur, Jonathan Katz, and Tjerand Silde. Two-round threshold lattice-based signatures from threshold homomorphic encryption. In *PQCrypto*, 2024.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, 2010.

MBH23. Christian Mouchet, Elliott Bertrand, and Jean-Pierre Hubaux. An efficient threshold access-structure for RLWE-based multiparty homomorphic encryption. *J. Cryptol.*, 2023.

MCPT24. Christian Mouchet, Sylvain Chatel, Apostolos Pyrgelis, and Carmela Troncoso. Helium: Scalable MPC among lightweight participants and under churn. In *CCS*, 2024.
OT25. Hiroki Okada and Tsuyoshi Takagi. Low communication threshold FHE from standard (module-)LWE. In *ASIACRYPT*, 2025.
PS24. Alain Passelègue and Damien Stehlé. Low communication threshold fully homomorphic encryption. In *ASIACRYPT*, 2024.
Sha79. Adi Shamir. How to share a secret. *Commun ACM*, 1979.
SSTX09. Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, 2009.

A Discussion on the Ebrahimi-Yadav Construction

In [EY24], the authors construct a ThFHE scheme in the asynchronous setting that achieves a strong simulation security notion. They do it by combining the scheme from [BGG$^+$18] with key-homomorphic pseudorandom functions [BLMR15]. The authors focus on the construction relying on linear secret sharing schemes with binary reconstruction coefficients, but, as they mention, it could be generalized to rely on Shamir secret sharing using the same denominator clearing technique as in [BGG$^+$18].

The main modification in their scheme is to add a mask generated with a key-homomorphic PRF to the partial decryption share of a party P_i, so that it takes the form:

$$\mathsf{p}_i = a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}} + r_i^{\mathsf{ct}} \ ,$$

where $r_i^{\mathsf{ct}} := F_{k_i}(\mathsf{ct})$, and where k_i is a secret share of the special key 0, which should satisfy $F_0(\mathsf{ct}) = 0$ for all ct. Thanks to the key-homomorphic property of F, recombination of the shares translates to recombination of the PRF key. The recombination of masks yields

$$\sum_{i \in \mathcal{S}} \lambda_i^{\mathcal{S}} r_i^{\mathsf{ct}} = F_{\sum_{i \in \mathcal{S}} \lambda_i^{\mathcal{S}} k_i}(\mathsf{ct}) = F_0(\mathsf{ct}) = 0 \ ,$$

which makes the masking term vanish.

One may think that their construction could provide a stg-SIM-secure target-decryptors ThFHE by integrating reconstruction coefficients in partial shares, e.g., as:

$$\mathsf{p}_i^{\mathcal{S}} = \lambda_i^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}} + F_{\lambda_i^{\mathcal{S}} \cdot k_i}((\mathsf{ct}, \mathcal{S})) \ .$$

However, this construction is actually insecure. The reason is that an attacker corrupting a set $\mathcal{S}_{\mathsf{cor}}$ of $t-1$ parties learns $t-1$ shares of the PRF key, but the PRF key has no entropy since it is fixed to 0 to ensure correctness after recombination. As any set of t PRF key shares leads to key 0, such an attacker can reconstruct the PRF key shares from all users. Hence, the attacker can compute the masking term $F_{\lambda_i^{\mathcal{S}} \cdot k_i}((\mathsf{ct}, \mathcal{S}))$ for any user i: indeed, it knows all quantities involved in this computation. After removing the mask, one recovers the scheme from [MBH23], and the attacker can use the attack from Section 4

to recover the secret key share sh_i of any party i. Overall, this construction is not even wk-IND-CPA-secure.

Our construction from Section 5 has a similar flavor, as we also use PRFs to mask $a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}}$. However, our masking technique achieves security, while using key-homomorphic PRFs does not in the synchronous setting. We emphasize that the above attack does not contradict any statement made in [EY24]: that work focuses on the asynchronous setting and makes no claim regarding the synchronous setting.

Conversely, our masking technique does not apply to the asynchronous setting, as we rely on the knowledge of the set of decryptors to generate our masks.

B Context-Dependent ThFHE

A recent work by Boneh et al. [BBN+25] introduced the notion of context-dependent threshold public-key encryption as an extension of threshold PKE. This notion adds an input dc to the partial decryption algorithm, called the *decryption context*. Concretely, a partial decryption is of the form:

$$\mathsf{p}_i^{\mathsf{dc}} \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathsf{dc}) \ .$$

To decrypt, one must collect t partial decryption shares from different parties for a matching decryption context dc. The authors define a security model that requires that, as long as the attacker does not collect t partial decryption shares (via corruption or partial decryption queries) for the same context, it learns nothiing about the plaintext underlying a ciphertext. The proposed security model is indistinguishability-based, and the work only considers PKE rather than FHE, but the guarantees are similar to our stg-SIM model.

While context-dependent ThFHE has not been defined explicitly, their definition extends to ThFHE and target-decryptors ThFHE, and our construction readily yields a context-dependent target-decryptors ThFHE via the following tweak: simply extend the PRF domain to include the decryption context dc as an additional input to the PRF when computing $\mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S}, \mathsf{dc})$.[4] Our analysis carries over to this setting without any modification beyond the syntax of PartDec.

C Increasing the Threshold for Decrypting a Ciphertext

In our construction, partial decryption outputs can be seen as containing two independent components: (1) the term $\lambda_i^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}}$, which contributes to decrypting ct and (2) the mask $r_i^{\mathcal{S},\mathsf{ct}}$, which prevents learning information about component (1) except by combining it with every other share required for decryption.

[4] One might additionally hash the decryption context dc with a collision-resistant hash function to fix the input length if necessary.

The mask is defined as the sum of PRF evaluations on $(\mathsf{ct}, \mathcal{S})$ over all keys held by party P_i and shared with another party in $\mathcal{S}$, i.e., keys $k_{i,j}$ or $k_{j,i}$ for $j \in \mathcal{S}$. Yet, we could analogously define masks for sets of decryptors $\mathcal{S}^*$ with $t' > t$ decryptors, and split $\mathcal{S}^*$ as $\mathcal{S}^* = \mathcal{S} \cup \mathcal{S}'$ with $\mathcal{S}$ containing t parties and $\mathcal{S}'$ containing $t' - t$ parties.

It is possible to enforce that decryption of a ciphertext $\mathsf{ct} = (a, b)$ requires a larger set of decryptors $\mathcal{S}^* = \mathcal{S} \cup \mathcal{S}'$ with $|\mathcal{S}| = t$. Decryptors in $\mathcal{S}$ produce partial decryptions of the form:

$$\mathsf{p}_i^{\mathcal{S}^*} = \lambda_i^{\mathcal{S}} \cdot a \cdot \mathsf{sh}_i + e_i^{\mathsf{smdg}} + r_i^{\mathcal{S}^*,\mathsf{ct}} \ ,$$

while decryptors in $\mathcal{S}'$ only contribute:

$$\mathsf{p}_{i'}^{\mathcal{S}^*} = r_{i'}^{\mathcal{S}^*,\mathsf{ct}} \ .$$

Correctness follows by inspection. Further, our security analysis extends to this setting: as long as the attacker corrupts no more than $t - 1$ parties, it does not learn the message underlying ct unless it learns all partial decryptions of the above form for all uncorrupted parties in $\mathcal{S}^* \setminus \mathcal{S}_{\mathsf{cor}}$. Even if $|\mathcal{S}^*| \gg t$, the attacker still learns nothing, even though many more than t partial decryptions of ct are revealed.

Of course, if the attacker corrupts t parties, it can always decrypt ct by simply reconstructing the FHE secret key and using it to decrypt.