Low Communication Threshold Fully Homomorphic Encryption

Alain Passelègue and Damien Stehlé

CryptoLab Inc., Lyon, France

Abstract. This work investigates constructions of threshold fully homomorphic encryption with low communication, i.e., with small ciphertexts and small decryption shares. In this context, we discuss in detail the technicalities for achieving full-fledged threshold FHE, and put forward limitations regarding prior works, including an attack against the recent construction of Boudgoust and Scholl [ASIACRYPT 2023]. In light of our observations, we generalize the definition of threshold fully homomorphic encryption by adding an algorithm which allows to introduce additional randomness in ciphertexts before they are decrypted by parties. In this setting, we are able to propose a construction which offers small ciphertexts and small decryption shares.

1 Introduction

Fully homomorphic encryption (FHE) [Gen09] allows to perform arbitrary computations on encrypted data. It has found numerous applications in cryptography. One of the vanilla applications is the delegation of heavy computation to a server: by encrypting data to the server, the server can perform the computation homomorphically on encrypted data to get an encryption of the result without learning any information about the raw input data nor the result. Yet, the data owner, who owns the FHE decryption key, can decrypt the evaluated ciphertext to get the plain result of the computation. Threshold fully homomorphic encryption [AJL$^+$12,BGG$^+$18] extends FHE in the multi-client setting: the decryption key is split between N parties so that to decrypt a ciphertext, each party must partially decrypt the ciphertext using its share of the key.[1] The plaintext is recovered by combining these partial decryptions. Threshold FHE has tremendous applications, including multi-party computation, universal thresholdizer, and delegation of computation over private data owned by multiple parties. In the latter scenario, each party can encrypt its data and upload it on a server. Anyone can ask the server to perform (homomorphic) computation over the joint data; yet, to learn the result of a computation, all parties must jointly decrypt the result, which allows parties to manage access to their data by possibly refusing decryption depending on the evaluated function.

[1] We focus on the N-out-of-N case of threshold FHE, in which all N parties must participate during decryption.

The most basic idea to turn an FHE scheme into a threshold scheme is as follows: FHE schemes have ciphertexts of the form $(\mathbf{a}, b)$ where $\mathbf{a} \in \mathbb{Z}_q^n$, and where $b = -\mathbf{a}^\intercal \mathbf{s} + \mu + e$ with $\mathbf{s} \in \mathbb{Z}_q^n$ being the secret key, μ the underlying plaintext, and e a small error term.[2] Decrypting such a ciphertext using $\mathbf{s}$ is done by computing $\mathbf{a}^\intercal \mathbf{s}$ and adding it to b to recover $\mu + e$ which is close to μ.[3] Transforming such a scheme into a threshold variant can be done by splitting the secret key $\mathbf{s}$ as $\mathbf{s} = \mathbf{s}_1 + \cdots + \mathbf{s}_N \bmod q$ and giving one share $\mathbf{s}_i$ of the key to each user P_i [BD10]. Decryption is then replaced by two algorithms $(\mathsf{PartDec}, \mathsf{FinDec})$. The first one is run by each party using its share of the key to compute a partial decryption share of the ciphertext, while the second one combines all partial decryption shares to recover the actual plaintext. At a high level, $\mathsf{PartDec}$ consists in computing $\mathsf{p}_i := \mathbf{a}^\intercal \mathbf{s}_i$, and $\mathbf{a}^\intercal \mathbf{s}$ is reconstructed in FinDec by adding all p_i's; $\mu + e$ is then recovered by computing $\mathbf{a}^\intercal \mathbf{s} + b$ as in the non-threshold case.

Obviously, this approach is not secure: each of P_i's partial decryption $\mathbf{a}^\intercal \mathbf{s}_i$ of a ciphertext $(\mathbf{a}, b)$ provides a linear equation in $\mathbf{s}_i$, therefore $\mathbf{s}_i$ can be recovered by Gaussian elimination given sufficiently many partial decryptions. An alternative way to see the problem is to observe that parties recover $\mu + e$ when decrypting, and then, if μ is known, they recover e. Given μ, e, and a ciphertext $(\mathbf{a}, \mathbf{a}^\intercal \mathbf{s} + \mu + e)$, one can then recover a linear equation $(\mathbf{a}, \mathbf{a}^\intercal \mathbf{s})$ in $\mathbf{s}$ and recover the secret key via Gaussian elimination. While this is not an issue in the non-threshold setting since decryption requires $\mathbf{s}$ anyway, the global secret key $\mathbf{s}$ should remain hidden to parties in the threshold setting. Hence, it is crucial that partial decryptions hide the value of the error term of the ciphertext e.

The solution proposed in [AJL+12,BGG+18] is to add a noise term $\mathfrak{d}_i$ to p_i, i.e., defining $\mathsf{p}_i := \mathbf{a}^\intercal \mathbf{s}_i + \mathfrak{d}_i$ where $\mathfrak{d}_i$ is an independent error term. Denoting $\mathfrak{d} := \sum_{i \in [N]} \mathfrak{d}_i$, the combination of partial shares during FinDec then allows to recover $\mu + e + \mathfrak{d}$ instead $\mu + e$. Regarding correctness of decryption, this is still fine as long as $e + \mathfrak{d}$ is small compared to μ and, hopefully, adding this error term now guarantees security by hiding e and therefore $\mathbf{s}$. The main question is then: *what is the minimal magnitude for $\mathfrak{d}_i$'s in order to guarantee security of the construction?* Ideally, one wants $\mathfrak{d}_i$ to be as small as possible, since the larger it gets, the larger one must set other parameters of the scheme to ensure correctness (and also to ensure security, which decreases as the ratio q/e increases).

Prior works. Early works have shown that adding an exponential noise term allows to guarantee the security of the scheme. Indeed, consider $\mathfrak{d}_i$ of magnitude $\Omega(2^\lambda B)$ where λ denotes the security parameter and B is a bound on the magnitude of the error term e of the ciphertext $(\mathbf{a}, \mathbf{a}^\intercal \mathbf{s} + \mu + e)$ to be decrypted.

[2] The most efficient FHE instantiations actually rely on Ring-LWE, i.e., replace $\mathbf{a}$, $\mathbf{s}$, μ and e by ring elements a, s, μ and e. We use LWE notations in the introduction.

[3] The plaintext μ can be a scaled version of a message. An additional rounding operation removes the error term for exact schemes (e.g., BGV [BGV12], B/FV [Bra12,FV12], DM/CGGI [DM15,CGGI16], or discrete versions of CKKS [DMPS24,BCKS24,BKSS24]), while approximate schemes (CKKS [CKKS17]) keep the approximate result $\mu + e$, seeing e as part of the approximation error of the computation over $\mathbb{R}$ or $\mathbb{C}$.

Then, the distribution of $\mu + e + \mathfrak{d}$ is statistically close to that of $\mu + \mathfrak{d}$, which is independent of e. Security follows: even if parties $P_1, \ldots, P_{N-1}$ get corrupted by an attacker, the last partial decryption share p_N of P_N allows the attacker to recover $\mu + e + \mathfrak{d}$, which is statistically close to $\mu + \mathfrak{d}$. The latter can be sampled publicly given the result μ of the decryption. Hence, the attacker learns statistically nothing about e and therefore $\mathbf{s}$, or equivalently $\mathbf{s}_N$, remains hidden.

While this provides security, exponential noise flooding induces a significant overhead in terms of efficiency, since the scheme must be correct even for exponentially large decryption error terms. Notably, the partial decryption shares p_i's include an exponential error term and must hence be very large, leading to high communication cost between parties. Consequently, several recent works have tried to reduce the magnitude of $\mathfrak{d}$ in order to improve the efficiency of the construction. The main work in this area is a recent paper by Boudgoust and Scholl [BS23a], which aims to build threshold FHE using a security analysis based on the Rényi divergence. The Rényi divergence has proven to be a powerful tool to reduce the magnitude of noise terms required in security analyses in the context of lattice-based signatures [BLL+18]. Yet, for primitives whose security is based on indistinguishability-based games such as FHE, the Rényi divergence has not been very successful. This contrasts with signatures, whose unforgeability security notion is a search-based game.

Two other recent works [CSS+22,DWF22] have also attempted to provide constructions relying on analysis based on the Rényi divergence in order to reduce the magnitude of the noise term added during partial decryption. It turns out that none of these works provides a valid solution to the above problem. In [DWF22], the authors only claim an extremely weak security statement, where security holds only for a single partial decryption. In [CSS+22], the authors propose a security analysis based on so-called public sampleability and claim security for threshold FHE based on DM/CGGI with polynomially many decryption queries. The security claim is again rather weak, since the adversary is selective (the adversary must declare its encryption and evaluation queries before seeing challenge ciphertexts). Furthermore, their argument for public sampleability (see the proof of Theorem 2 in [CSS+22]) relies on a distribution for the randomness r which can depend on the error underlying an evaluated ciphertext. For public sampleability, this distribution should be independent of the challenge bit, but the error underlying an evaluated ciphertext could depend on the plaintexts involved in the computation (hence on the challenge bit) even though the result of the computation is independent of the challenge bit.

In [DDK+23], the authors provide an intermediate step to efficient threshold FHE, based on DM/CGGI, by proposing the following approach: practical parameters are used to encrypt and evaluate computation, but before partial decryption is performed, a ciphertext is passed through a "switch-n-squash" process. The latter bootstraps a ciphertext to sufficiently large parameters while keeping the noise small, so that it is possible to flood the noise during threshold decryption of the resulting (large) ciphertext. This allows to have half of the communication reduced (communication towards the server) but decryption shares remain large due to (exponential) noise flooding.

Finally, note that theoretical solutions to threshold FHE exist using generic MPC techniques (e.g., [BJKL21,Shi22] implies an N-out-of-N threshold FHE with $\mathsf{poly}(\lambda)$-size decryption shares), but none of these results achieves practical efficiency. However, focusing only on threshold PKE instead of FHE, a recent work by Micciancio and Suhl [MS23] provides an elegant solution (in the N-out-of-N threshold setting) from LWE via a careful analysis of the noise distributions. Our main construction follows a similar analysis as theirs, but for fully homomorphic encryption.

1.1 Our Contributions

First, we remark that only adding a small error term during partial decryption cannot succeed for some classical schemes (e.g., B/FV or CKKS). This is due to the error distribution during homomorphic computation and has already been observed in the context of $\mathsf{IND\text{-}CPA\text{-}D}$ security for CKKS [LMSS22]. Hence, to circumvent this issue, a natural direction is to further rely on techniques to sanitize the noise during homomorphic computation, such as relying on circuit-private FHE, as proposed in [BS23a]. Unfortunately, we show that the transform from [BS23a] fails to provide threshold security, by proposing a circuit-private FHE scheme which is insecure when plugged into their transform. In addition, relying on sanitization techniques (which induce randomized evaluation) rises a challenging question about the model: security of sanitization is guaranteed based on the (private) internal randomness of evaluation. Relying on randomized evaluation in the threshold setting then questions which party does the randomized evaluation, and what can we hope from this approach if the party doing the evaluation (and therefore knowing its internal randomness) is corrupted?

Based on these observations, we extend the definition of threshold FHE by adding an additional algorithm which allows an uncorrupted party (we assume it is a server) to process on evaluated ciphertexts before parties run partial decryption. This randomized operation, termed $\mathsf{ServerDec}$, uses randomness unknown to parties. Our definition matches the standard threshold FHE definition when $\mathsf{ServerDec}$ is deterministic (or, equivalently, void).

We then propose a construction guaranteeing security in this setting. Our main construction is round optimal, and relies on two flooding steps: $\mathsf{ServerDec}$ adds an exponential noise to ciphertexts after evaluation, but also compresses them. Hence, ciphertexts fed to partial decryptions are small, and we prove that adding a very small noise in $\mathsf{PartDec}$ is sufficient for security. In the context of delegation of computation on a trusted server, this means that the communication from the server to the parties as well as between parties is small, even if exponential noise flooding is used on the server side.

We complete our work by providing a few more contributions in the Appendices. First, in Appendix B we propose a protocol designed for threshold delegation of computation over private data. It is not a threshold FHE scheme properly speaking: our protocol requires an additional round before partial decryption. A specific party (distinct from the server, but possibly one of the partial decryptors) needs to process the evaluated ciphertext before the other parties can run partial decryption. On the positive side, our protocol is only based on circuit-private (non-threshold) FHE and threshold public-key encryption. Hence it can avoid any (exponential) noise flooding by relying on circuit-private transforms (e.g., [DS16,BPMW16,Klu22,BI22]). Second, in Appendix C, we describe connections between various security notions for FHE, notably an indistinguishability-based notion of threshold security, IND-CPA-D security, and a weak form of circuit-privacy.

1.2 Technical Overview

We start our technical overview by describing vulnerabilities in prior attempts to achieve threshold FHE with small partial decryption shares.

Our first observation is a direct generalization of known results in the context of the CKKS approximate FHE scheme. Recall that a CKKS ciphertext $(\mathbf{a}, b)$ with $b = \mathbf{a}^\intercal \mathbf{s} + \mu + e$ is decrypted by returning $\mu + e = b + \mathbf{a}^\intercal \mathbf{s}$. In [LM21], the authors show that decryption results can be exploited to mount so-called IND-CPA-D attacks, which simply exploit the fact that the decryption equation is a linear equation in $\mathbf{s}$. It is shown in [LMSS22] that adding a Gaussian noise $\mathfrak{d}$ to the decryption result $\mu + e$ allows to prevent such attacks, but the magnitude of the noise needs to be exponential in the security parameter. This was generalized to the threshold variant of CKKS in [KS23]. The attack exploits the fact that when multiplying two CKKS ciphertexts encrypting μ_0, μ_1 with error terms e_0, e_1, the error term of the resulting ciphertext is of the form $\mu_0 e_1 + \mu_1 e_0 + e'$ where $\|e'\| \ll \|\mu_0 e_1 + \mu_1 e_0\|$. Hence, the error distribution is highly-dependent on the underlying plaintexts. Two computations leading to the same result could therefore have vastly different error terms because they have different intermediate values, and hiding the bias during decryption requires to add an exponentially large noise term. We observe that in the context of threshold FHE, this attack does not exploit any specificity of CKKS. Since B/FV has identical noise propagation during homomorphic multiplications as CKKS, similar lower bounds on $\mathfrak{d}$ are also required to guarantee the security of threshold B/FV. Hence, it seems that, in general, only adding a small error term during partial decryption cannot be a solution to obtain threshold security. This motivates the use of sanitization, to make the noise of the result independent of the intermediate values.

An attack against the Boudgoust-Scholl transform. In [BS23a], the authors show that a circuit-private FHE scheme can be turned into a threshold FHE scheme satisfying a one-way security notion by secret-sharing the secret key between parties and adding an error term of magnitude growing with Q_D to partial decryption, where Q_D is an upper bound on the number of decryptions made. Note that Q_D can be much lower than exponential in the security parameter. This is made possible by a security analysis based on the Rényi divergence, thanks to the fact that one-way security is a search-based security game. Then, the authors claim that the one-way threshold FHE schemes can be turned into an IND-CPA threshold FHE schemes via a transform based on hard-core bit predicates. We refer the reader to Section 3 for details about the transform.

Unfortunately, we show that this transform fails to provide IND-CPA security of the resulting threshold FHE scheme. We construct a circuit-private FHE scheme which results in an insecure threshold FHE scheme once plugged into the transform. Let us now provide a simplified overview of the attack. Consider a (circuit-private, threshold) FHE scheme whose ciphertexts have the form $(\mathbf{a}, \mathbf{a}^{\intercal}\mathbf{s}+\mu+e)$. The adversary's goal is to distinguish an encryption of μ_0 from an encryption of μ_1 given access to a (partial) decryption oracle. Let $(\mathbf{a}^*, b^*)$ denote the challenge ciphertext, where $b^* = \mathbf{a}^{*\intercal}\mathbf{s} + \mu^{(\beta)} + e^*$, with $\beta \in \{0,1\}$ denoting the challenge bit. Further assume that it is possible, via the circuit-private evaluation, to transform a ciphertext encrypting a plaintext μ into another ciphertext encrypting the same plaintext, but whose error term e' is now of the form $\mu^{\intercal}\mathbf{s}$. We emphasize that this does not contradict circuit-privacy.[4] Our attack exploits the fact that the adversary can ask decryption of ciphertexts whose underlying plaintext depends on $\mathbf{a}^*$. Specifically, our attacker submits an encryption query whose plaintext encodes the most significant bits $[\mathbf{a}^*]$ of $\mathbf{a}^*$, applies the above circuit-private mechanism, and then asks for decryption of the result. Ignoring the small noise term added during partial decryption, the adversary then recovers an error term of the form $[\mathbf{a}^*]^{\intercal}\mathbf{s}$, and putting the bits in the appropriate slots, it can then subtract the most significant bits of $\mathbf{a}^{*\intercal}\mathbf{s}$ from b^* to guess the value of β, leading to a distinguishing attack and contradicting the security of the transform. The attack being adaptive (it involves a query that depends on the challenge ciphertext), it might be possible to prove the security of the transform in a selective case, requiring the adversary to declare all its encryption and evaluation queries before seeing the challenge ciphertext. However, this model is much weaker than ours and than the model considered in [BS23a]. (At the time of writing, the current eprint version [BS23b] of [BS23a] considers selective security, the update having been made based on our notification of a proof flaw to the authors.)

A generalized definition of threshold FHE. Given our first observation, it seems that sanitizing evaluated ciphertexts such that the error term is independent of the underlying plaintexts is a good start. This can be done by randomizing evaluation (e.g., using circuit-private evaluation), but it poses a definitional issue: in threshold FHE, any party can be corrupted, hence if a party performs the evaluation and gets corrupted, the randomness used during the evaluation is revealed to the adversary. In this case, there is no randomness in the evaluation and one cannot rely on it to argue about security, falling back to the prior issues.[5]

[4] One could notice that e' has no reason to be small or might not even be properly defined. We assume this is not the case, only for simplifying the overview. Our actual attack avoids this issue.

[5] Note that relying on randomized evaluation also induces issues for some applications of threshold FHE to threshold cryptography. We provide an example with threshold

To circumvent this issue, we modify the definition of threshold FHE by adding an algorithm termed $\mathsf{ServerDec}$. We define a threshold FHE scheme as a tuple $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{ServerDec}, \mathsf{PartDec}, \mathsf{FinDec})$ where KeyGen, Enc, Eval, $\mathsf{PartDec}$, and FinDec are the usual threshold FHE algorithms. $\mathsf{ServerDec}$ is an additional, public-key, randomized algorithm, which can be performed on fresh or evaluated ciphertexts, and before partial decryption. It returns a ciphertext which fits the input format of $\mathsf{PartDec}$. We suppose that $\mathsf{ServerDec}$ is executed by an uncorrupted party (e.g., the server) and uses randomness that remains unknown to all parties (and therefore to the adversary). We recover the prior syntax of threshold FHE when $\mathsf{ServerDec}$ is void (or equivalently, when it is deterministic). Note that, assuming homomorphic evaluation is performed on an uncorrupted server as for delegation of computation, $\mathsf{ServerDec}$ can be performed directly following the Eval algorithm by the server. It may seem that it could then be integrated directly in Eval algorithm, but adding an explicit algorithm increases flexibility: for instance, $\mathsf{ServerDec}$ might convert ciphertext into a format which prevents further homomorphic evaluation. Also, adding an explicit $\mathsf{ServerDec}$ algorithm allows Eval to possibly remain deterministic. Finally, it allows to identify precisely the sources of security. Considering an adversary which corrupts all first $N-1$ users, the security is guaranteed thanks to the remaining uncorrupted randomness, i.e., (1) the internal randomness of $\mathsf{ServerDec}$, and (2) the randomness held by P_N (its share $\mathbf{s}_N$ of the key and its internal randomness of $\mathsf{PartDec}$).

Speaking of security, we define a simulation-based security notion which closely follows prior notions from [BGG+18]. It is actually slightly stronger: adaptive queries and multi-hop evaluations are permitted. In short, our security notion requires that an adversary corrupting at most $N-1$ parties should not learn any valuable information beyond the value of the underlying plaintext. This is modeled by the existence of an efficient simulator which takes as input information available to the adversary (the corrupted shares of the secret key as well as the plaintext underlying the ciphertext to be decrypted), and returns a simulated ciphertext as well as simulated partial decryption shares for all parties. We further require that the distribution of the simulation is indistinguishable from that of the real ciphertext produced by $\mathsf{ServerDec}$ and of the honestly generated partial decryption shares.

Double-flood-and-round threshold FHE. Our main result is a construction of threshold FHE with low communication. We call our technique *Double-Flood-and-Round.* consider a generic FHE scheme whose ciphertexts have the above form $(\mathbf{a}, \mathbf{a}^\intercal \mathbf{s} + \mu + e)$ over $\mathbb{Z}_Q$ for some (exponentially) large modulus $Q = p \cdot q_{\mathsf{dec}}$

signatures. The construction from [BGG+18] of threshold signature from threshold FHE requires parties to homomorphically evaluate a deterministic signing algorithm, where the signing key is encrypted, and to reveal partial decryptions of the resulting ciphertext as partial signatures. For correctness, it is crucial that all parties evaluate the same signature (which is why it is chosen deterministic), but also the same ciphertext: randomized evaluation is not an option. However, an analysis based on Rényi divergence shows that a noise growing with Q_D suffices in this setting [ASY22].

with $p = \Omega(2^\lambda)$ and $q_{\mathsf{dec}} = \mathsf{poly}(\lambda)$. The key generation, encryption, and evaluation algorithms are directly inherited from the underlying FHE scheme, except that the secret key $\mathbf{s}$ is additively secret shared as $\mathbf{s} = \mathbf{s}_1 + \cdots + \mathbf{s}_N \bmod q_{\mathsf{dec}}$. Our technique is to first rely on exponential noise flooding to sanitize evaluated ciphertexts before partial decryption. This is done by the $\mathsf{ServerDec}$ algorithm. It takes a ciphertext $(\mathbf{a}, b) \in \mathbb{Z}_Q^n \times \mathbb{Z}_Q$ and returns a ciphertext $(\mathbf{a}', b') \in \mathbb{Z}_{q_{\mathsf{dec}}}^n \times \mathbb{Z}_{q_{\mathsf{dec}}}$ with:

$$\mathbf{a}' = \left\lfloor \frac{1}{p} \cdot \mathbf{a} \right\rceil_{\sigma_0} \quad , \quad b' = \left\lfloor \frac{1}{p} \cdot (b + \mathfrak{E}) \right\rceil_{\sigma_1} \quad .$$

where $\lfloor \cdot \rceil_\sigma$ denotes a randomized Gaussian rounding, which on input $x \in \mathbb{R}$ returns an element from $\mathcal{D}_{\mathbb{Z},x,\sigma}$, and where $\mathfrak{E}$ is an exponentially large Gaussian noise term. Via standard noise flooding, the error term $\mathfrak{E}$ statistically hides the error term from b. Moreover, thanks to the rescaling to $q_{\mathsf{dec}} = \mathsf{poly}(\lambda)$, the ciphertext $(\mathbf{a}', b')$ sent to parties to be partially decrypted is only $(n+1)\log(q_{\mathsf{dec}})$-bit long. $\mathsf{PartDec}$ and FinDec then follow the same design as before.

We are able to prove that adding a very small amount of noise (whose magnitude is even independent of the number of decryptions Q_D) during partial decryption suffices to guarantee security. Specifically, a ciphertext $(\mathbf{a}', b')$ returned by $\mathsf{ServerDec}$ is of the form $(\mathbf{a}', \mathbf{a}'^{\intercal}\mathbf{s} + \mu + e')$, where e' has the form $\left\lfloor \frac{1}{p} \cdot (\mathbf{r}_0^{\intercal}\mathbf{s} + \mathfrak{E}) \right\rceil_{\sigma_1}$ with $\mathbf{r}_0$ denoting the (Gaussian) rounding error of $\mathbf{a}$, i.e., $\mathbf{r}_0 = \mathbf{a} - p \cdot \mathbf{a}'$. Therefore, following a similar approach as the proof of [BLP+13, Lemma 3.5], one can show that e' is statistically close from a Gaussian distribution whose standard deviation is $\sqrt{(\sigma_0\|\mathbf{s}\|)^2 + (\sigma_{\mathsf{flood}}/p)^2 + \sigma_1^2}$. Assuming $\|\mathbf{s}\|$ is publicly known, this distribution is publicly sampleable. Thanks to this clean distribution of e', the rest of the security analysis is analogous to the recent proof of security of lattice-based threshold public-key encryption from [MS23].

A partial decryption of $(\mathbf{a}', b')$ computed by party P_N owning $\mathbf{s}_N$ is of the form $\mathbf{a}'^{\intercal}\mathbf{s}_i + \mathfrak{d}_i$. Sampling $\mathfrak{d}_N$ from $\mathcal{D}_{\mathbb{Z},\eta}$, we then obtain that the view of an adversary corrupting all parties $P_1, \ldots, P_{N-1}$ (and therefore $\mathbf{s}_1, \ldots, \mathbf{s}_{N-1}$) is a triple $(\mathbf{a}', \mathbf{a}'^{\intercal}\mathbf{s} + \mu + e', \mathbf{a}'^{\intercal}\mathbf{s}_N + \mathfrak{d}_N)$, and adding $\mathbf{a}'^{\intercal} \sum_{i\in[N-1]} \mathbf{s}_N$ to the third term, it is then a triple of the form:

$$(\mathbf{a}', \mathbf{a}'^{\intercal}\mathbf{s} + \mu + e', \mathbf{a}'^{\intercal}\mathbf{s} + \mathfrak{d}_N) \ ,$$

with $e' \sim \mathcal{D}_{\mathbb{Z},\sqrt{(\sigma_0\|\mathbf{s}\|)^2 + (\sigma_{\mathsf{flood}}/p)^2 + \sigma_1^2}}$ and $\mathfrak{d}_N \sim \mathcal{D}_{\mathbb{Z},\eta}$. Assuming LWE, we prove the above distribution is computationally indistinguishable from a triple:

$$(\mathbf{a}', b', b' + \mathfrak{h}) \ ,$$

where $\mathfrak{h} \leftarrow \mathcal{D}_{\sigma_{\mathfrak{h}}}$ with $\sigma_{\mathfrak{h}} := \sqrt{(\sigma_0\|\mathbf{s}\|)^2 + (\sigma_{\mathsf{flood}}/p)^2 + \sigma_1^2 + \eta^2}$. The latter distribution is publicly sampleable if $\|\mathbf{s}\|$ is known, which does not hurt the analysis.

We further note that the partial decryption shares can be rounded, in order to lower communication (we do not use this for security). This explains why the construction is called double-round-and-flood. As a final remark, note that our input FHE ciphertexts are not compact by default, as they are defined over a large modulus Q. This problem is solved by relying on transciphering, e.g., using [BCK⁺23]. Hence, all communications are small, and only the steps run by the server are dealing with large ciphertexts.

Additional contributions. We provide a few more contributions in the Appendices. First, in Appendix B, we propose an alternative protocol for delegation of computation in a threshold setting which combines threshold public key encryption with circuit-private FHE. The idea is the following: users generate a threshold PKE public key and shares of the secret key. A special party, termed the *transcryptor*, which could be a trusted third party or one of the users, generates a pair of keys for a (circuit-private) FHE scheme and keeps the secret key for itself while revealing the public key to the server and all users. We assume that the transcryptor and the server are not colluding and that communications between every party is done via secure channels (e.g., using authenticated symmetric encryption). The server receives data from each user, encrypted under the FHE public key, via secure channels. Then, to perform a computation C over data $\mu_1, \ldots, \mu_N$, the server homomorphically evaluates $\mathsf{ThEnc}_{\mathsf{tpk}} \circ \mathsf{C}$ on the data, where ThEnc denotes the encryption algorithm of the threshold PKE scheme and tpk its public key. The result is an FHE encryption of a threshold PKE encryption of $\mathsf{C}(\mu_1, \ldots, \mu_N)$. Then, this ciphertext is sent to the transcryptor, which decrypts it using the FHE secret key, and broadcasts $\mathsf{ThEnc}_{\mathsf{tpk}}(\mathsf{C}(\mu_1, \ldots, \mu_N))$ to all users. This step adds a communication round compared to approaches based on threshold FHE in which the outputs of $\mathsf{ServerDec}$ computed by the server can be directly decrypted by all parties: it is not possible to avoid this additional round since the party (here, the server) performing evaluation must be independent of the FHE secret key holder. The protocol completes decryption of the result by having all parties jointly decrypting the threshold PKE ciphertext using their shares of the decryption key.

Assuming circuit-privacy of the underlying FHE scheme and simulation security of the underlying threshold PKE scheme, this protocol achieves simulation security. This is proven by simulating the view of an adversary by simulating $\mathsf{ThEnc}_{\mathsf{tpk}}(\mathsf{C}(\mu_1, \ldots, \mu_N))$ using the threshold PKE simulator, and by replacing the ciphertext computed by the server with a fresh FHE encryption of this simulated threshold PKE ciphertext. Using circuit-private FHE directly allows to avoid the use of exponential noise flooding when circuit-privacy is achieved by mechanisms such as those from [DS16,BPMW16,Klu22,BI22].

We also complete the paper with discussions about various advanced security notions for FHE, including threshold security, IND-CPA-D security, and circuit-privacy, and provide connections between these notions. These results are detailed in Appendix C.

Additional related work. In [ASY22], the authors construct threshold signatures based on threshold FHE techniques, again by relying on an analysis based on the Rényi divergence to reduce the magnitude of the added noise. We emphasize that the authors do not build a threshold FHE scheme, but only use FHE as a building block in a threshold flavour as part of their threshold signature construction. The reduction does not rely on the security of threshold FHE:

the authors directly prove unforgeability (a search-based game) of the threshold signature by arguing that a small noise is sufficient.

Acknowledgments. We thank Katharina Boudgoust and Peter Scholl for fruitful discussions, as well as Intak Hwang, Seonghong Min and Yongsoo Song for helpful feedback on a prior version of this work.

2 Preliminaries

Notation. For any integer $N \geq 1$, we let $[N]$ denote the set $\{1, \dots, N\}$. Vectors and matrices are written in bold letters. Vectors are column vectors. For a vector $\mathbf{x}$, we let $\mathbf{x}^\intercal$ denote its transpose. Given a finite set S, we let $U(S)$ denote the uniform distribution over S. The notation log refers to the natural logarithm. We use the notations $\mathsf{negl}(\lambda) = \lambda^{-\omega(1)}$ and $\mathsf{poly}(\lambda) = \lambda^{O(1)}$, where λ refers to the security parameter. For X, Y two distributions over a countable set Ω, the statistical distance between X and Y is defined as $\Delta(X, Y) := \frac{1}{2} \sum_{\omega \in \Omega} |\Pr[X = \omega] - \Pr[Y = \omega]|$.

For an integer $x \in \mathbb{Z}$, a modulus $q > 0$, and an integer $N > 0$, we let Share denote the standard additive secret sharing algorithm which takes as input (x, N, q), samples $(x_1, \dots, x_{N-1}) \leftarrow U(\mathbb{Z}_q^{N-1})$, and returns $(x_1, \dots, x_N)$ with $x_N = x - \sum_{i \in [N-1]} x_i \bmod q$.

2.1 Gaussian Distributions

Definitions. For an integer $n > 0$ and $\sigma > 0$, we define the n-dimensional Gaussian function $\rho_\sigma : \mathbb{R}^n \to (0, 1]$ as:

$$\rho_\sigma(\mathbf{x}) := \frac{1}{\sigma^n} \exp\left(\frac{-\pi \|\mathbf{x}\|^2}{\sigma^2}\right) .$$

We say that a random variable X over $\mathbb{R}^n$ follows the Gaussian distribution of standard deviation σ and center $\mathbf{c} \in \mathbb{R}^n$, denoted $\mathcal{D}_{\mathbf{c},\sigma}$, if its density function is $\rho_X : \mathbf{x} \mapsto \rho_\sigma(\mathbf{x} - \mathbf{c})$. Similarly, a random variable X over $\mathbb{Z}^n$ follows the discrete Gaussian distribution of standard deviation parameter σ and center parameter $\mathbf{c}$ if the probability mass function of X is given by:

$$\Pr[X = \mathbf{x}] = \frac{\rho_\sigma(\mathbf{x} - \mathbf{c})}{\rho_\sigma(\mathbb{Z}^n)} .$$

We let $\mathcal{D}_{\mathbb{Z}^n,\sigma,\mathbf{c}}$ denote the n-dimensional discrete Gaussian distribution of standard deviation parameter σ and center parameter $\mathbf{c}$, and drop the index $\mathbf{c}$ if $\mathbf{c} = \mathbf{0}$. We also remark that $\mathcal{D}_{\mathbb{Z}^n,\sigma} = \mathcal{D}_{\mathbb{Z},\sigma}^n$. The definition generalizes to shifted supports $\mathbb{Z}^n - \mathbf{c}$. These distributions are efficiently sampleable for all σ (see, e.g., [BLP+13, Section 5.1]).

Gaussian rounding. In our main construction, we rely on randomized Gaussian roundings: for a standard deviation parameter $\sigma > 0$, we let $\lfloor \cdot \rceil_\sigma$ denote the

Gaussian rounding operation which, on input a value $x \in \mathbb{R}$ returns a sample from $\mathcal{D}_{\mathbb{Z},x,\sigma}$, or equivalently, samples z from $\mathcal{D}_{\mathbb{Z}-x,\sigma}$ and returns $x+z$. We extend it to vectors in a component-wise manner.

Elementary results. We prove the following smudging lemma in Appendix A.

Lemma 2.1. *Let $\sigma > 0$ and $c_0, c_1 \in \mathbb{Z}$. Then:*

$$\varDelta\left(\mathcal{D}_{\mathbb{Z},c_0,\sigma}, \mathcal{D}_{\mathbb{Z},c_1,\sigma}\right) \leq O\left(\frac{|c_0 - c_1|}{\sigma}\right) .$$

In particular, for $\lambda > 0$, $c \in \mathbb{Z}$ and $\sigma > \varOmega(c2^\lambda)$, we have $\varDelta(\mathcal{D}_{\mathbb{Z},\sigma}, \mathcal{D}_{\mathbb{Z},c,\sigma}) < 2^{-\lambda}$.

We finally recall the following results about lattice Gaussians. They are expressed in terms of the smoothing parameter $\eta_\varepsilon(\mathbb{Z}^n)$, defined, for arbitrary $\varepsilon > 0$ and integer $n \geq 1$, as the smallest $s > 0$ such that $\rho_{1/s}(\mathbb{Z}^n \backslash \{\mathbf{0}\}) \leq \varepsilon$. By [GPV08, Lemma 3.1], we know that $\eta_\varepsilon(\mathbb{Z}^n) \leq \sqrt{\log(2n(1+1/\varepsilon))/\pi}$.

Lemma 2.2. *[Reg05, Corollary 3.10] Let $n \geq 1$, $\mathbf{a}, \mathbf{s} \in \mathbb{R}^n$ and $\sigma, \psi > 0$. Assume that $(1/\sigma^2 + (\|\mathbf{s}\|/\psi)^2)^{-1/2} \geq \eta_\varepsilon(\mathbb{Z}^n)$ for some $\varepsilon < 1/2$. Then, the distribution of $\mathbf{x}^\intercal \mathbf{s} + e$ where $\mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z}^n + \mathbf{a},\sigma}$ and $e \leftarrow \mathcal{D}_\psi$ is at statistical distance at most 4ε from $\mathcal{D}_{\sqrt{(\sigma\|\mathbf{s}\|)^2+\psi^2}}$.*

Lemma 2.3. *[Pei10, Theorem 3.1] Let $n \geq 1$ and $\sigma, \psi > 0$ with $\sigma \geq \eta_\varepsilon(\mathbb{Z}^n)$ for some $\varepsilon < 1/2$. If sampling $\mathbf{x} \leftarrow \mathcal{D}_\psi$ and $\mathbf{y} \leftarrow \mathcal{D}_{\mathbb{Z}^n - x,\sigma}$, the distribution of $\mathbf{x} + \mathbf{y}$ is at statistical distance at most 8ε from the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^n,\sqrt{\sigma^2+\psi^2}}$.*

2.2 Hardness Assumptions

We first remind the standard LWE assumption.

Definition 2.4. *Let $n, m, q, \psi > 0$, and $\chi_{\mathbf{s}}$ denote a distribution over $\mathbb{Z}^n$. These parameters are function of the security parameter λ. The $\mathsf{LWE}_{n,m,q,\psi,\chi_{\mathbf{s}}}$ assumption states that the distributions*

$$(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \quad \text{and} \quad (\mathbf{A}, \mathbf{u})$$

are computationally indistinguishable, where $\mathbf{A} \leftarrow U(\mathbb{Z}_q^{m \times n})$, $\mathbf{s} \leftarrow \chi_{\mathbf{s}}^n$, $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\psi}$ and $\mathbf{u} \leftarrow U(\mathbb{Z}_q^m)$.

Our main construction relies on the following yaLWE assumption (yet another LWE assumption), which is implied by the standard LWE assumption, as we explain below. It combines the Reused-$\mathbf{A}$ LWE and Known-Norm LWE assumptions considered in [MS23].

Definition 2.5. *Let $n, m, q, \sigma, \eta > 0$ and $\chi_{\mathbf{s}}$ denote a distribution over $\mathbb{Z}^n$. These parameters are function of the security parameter λ. The $\mathsf{yaLWE}_{n,m,q,\sigma,\eta,\chi_{\mathbf{s}}}$ assumption states that the distributions*

$$(\mathbf{A}, \mathbf{As} + \mathfrak{e}, \mathbf{As} + \mathfrak{d}, \|\mathbf{s}\|) \quad \text{and} \quad (\mathbf{A}, \mathbf{u}, \mathbf{u} + \mathfrak{h}, \|\mathbf{s}\|)$$

are computationally indistinguishable, where $\mathbf{A} \leftarrow U(\mathbb{Z}_q^{m\times n})$, $\mathbf{s} \leftarrow \chi_{\mathbf{s}}^n$, $\mathfrak{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$, $\mathfrak{d} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\eta}$, $\mathbf{u} \leftarrow U(\mathbb{Z}_q^m)$ and $\mathfrak{h} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sqrt{\sigma^2+\eta^2}}$.

Lemma 2.6. *Let $n, m, q, \sigma_{\mathbf{s}}, \sigma_{\mathbf{e}}$ and $\eta > 0$, with $\sigma_{\mathbf{e}} \geq \sqrt{2}\sigma_{\mathbf{s}} \geq \Omega(\sqrt{\lambda + \log n})$ and $\sigma_s \leq \mathsf{poly}(\lambda)$. Assume that $\chi_{\mathbf{s}} = \mathcal{D}_{\mathbb{Z}^n,\sigma_{\mathbf{s}}}$. If the $\mathsf{LWE}_{n,m,q,\psi,\chi_{\mathbf{s}}}$ assumption holds for $\psi = (\sqrt{\sigma_{\mathbf{e}}^{-2} + \eta^{-2}})^{-1/2}$, then the $\mathsf{yaLWE}_{n,m,q,\sigma_{\mathbf{e}},\eta,\chi_{\mathbf{s}}}$ assumption holds.*

This essentially follows from [MS23, Corollary 3 and Lemma 9]. In the latter, the authors prove that $\mathsf{LWE}_{n,m,q,\psi,\chi_{\mathbf{s}}}$ implies the Reused-$\mathbf{A}$ $\mathsf{LWE}_{n,m,q,\sigma_s,\eta,\chi_{\mathbf{s}}}$ assumption. The latter assumption precisely corresponds to our $\mathsf{yaLWE}_{n,m,q,\sigma_{\mathbf{s}},\eta,\chi_{\mathbf{s}}}$ except for $\|\mathbf{s}\|$ which is kept secret. (Reused-$\mathbf{A}$ LWE considers $\chi_{\mathbf{s}} = U(\mathbb{Z}_q^n)$, but the analysis from [MS23] generalizes to arbitrary secret key distributions $\chi_{\mathbf{s}}$.) Then, Lemma 2.6 follows by the same observation as for Known-Norm LWE in the same work: any solver for the search variant of $\mathsf{yaLWE}_{n,m,q,\sigma_{\mathbf{s}},\eta,\chi_{\mathbf{s}}}$ yields a solver for the search variant of Reused-$\mathbf{A}$ $\mathsf{LWE}_{n,m,q,\sigma_{\mathbf{s}},\eta,\chi_{\mathbf{s}}}$, by first guessing $\|\mathbf{s}\|$ (this guess being correct with probability $1/\mathsf{poly}(\lambda)$ as long as $\|\mathbf{s}\| = \mathsf{poly}(\lambda)$); then one can rely on [MM11] to relate the search and decision variants. Note that [MS23, Lemma 9] requires the same distribution for the coordinates of $\mathbf{s}$ and $\mathbf{e}$. In our application (and in the statement of Lemma 2.6), we use larger a larger standard deviation parameter for the coefficients of $\mathbf{e}$ than for those of $\mathbf{s}$. Reducing the same-noise variant to the latter one is achieved by adding Gaussian noise to the second and third coordinates of the yaLWE sample. The sum of two discrete Gaussians is indeed very close to a discrete Gaussian (see [BF11, Lemma 4.12]).

3 Limitations from Prior Works

We start by discussing prior attempts to build threshold FHE with small partial decryption shares. Threshold FHE (and variants of it) has received a lot of attention in the last few years and some works (notably [BS23a,CSS+22]) claim to obtain efficient constructions based on an analysis relying on the Rényi divergence. We already pointed issues with [CSS+22] in the introduction and now focus on explaining technical issues with [BS23a]. We first discuss about the need for randomness before partial decryption happens to achieve threshold FHE with small partial decryption shares, then detail our analysis of [BS23a].

3.1 On the Need for Randomness Before Partial Decryption

For the approximate FHE scheme CKKS, lower bounds were proven in [LMSS22] regarding the amount of (Gaussian) noise to be added after decryption, in order to achieve IND-CPA-D security (see Appendix C for a precise definition) when the evaluation algorithm Eval of the scheme is deterministic. The authors proved that decryption needs to add a noise of magnitude $\Omega(2^{\lambda/4})$, and a similar lower bound was recently proven in the threshold setting (still for CKKS only) in [KS23, Theorem 12]. (The result is written in the case of multi-key FHE but extends to threshold FHE.) Our first observation is that these results do not rely on any specificity of CKKS, except on the fact that the noise after multiplication of two ciphertexts, encrypting μ_1, μ_2 and with noise terms e_1, e_2, is of the form $\mu_1 e_2 + \mu_2 e_1 + e'$ with $\|e'\| \ll \|\mu_1 e_2 + \mu_2 e_1\|$. This noise propagation is also that of the (exact) B/FV FHE scheme. The lower bound can then be extended to any threshold FHE scheme with such a format, and in particular to B/FV. This is not surprising, given that IND-CPA-D can be seen as a particular case of threshold FHE security (see discussion in Appendix C for details). In short, this is due to the fact that CKKS decryption corresponds to (noiseless) PartDec as defined earlier.

To obtain the lower bound, we consider four challenge queries of the form

$$\begin{aligned}(\mu_1^{(0)}, \mu_1^{(1)}) = (0, B),\ \ (\mu_2^{(0)}, \mu_1^{(1)}) = (0, B)\ \ ,\\ (\mu_3^{(0)}, \mu_3^{(1)}) = (0, B),\ \ (\mu_4^{(0)}, \mu_4^{(1)}) = (0, -B)\ \ .\end{aligned}$$

We then evaluate $\mu_1\mu_2 + \mu_3\mu_4$, which leads to an encryption of 0 (computed as $0^2 + 0^2$ or as $B^2 - B^2$ respectively, depending on the challenge bit β). The fact that intermediate plaintexts are 0 is one case and of magnitude B in the other case implies that the underlying error of the result is simply e' for $\beta = 0$, while it is of the form $Be + e'$ with e' small relative to Be for $\beta = 1$. This difference in magnitude can be further amplified by making not just 4 queries but $4t$ queries of the same form and evaluating $\sum_{i\in[t]}(\mu_{1,i}\mu_{2,i} + \mu_{3,i}\mu_{4,i})$. As explained in the introduction, partial decryptions allow to recover the error, up to the error added by each partial decryption. Hence, unless partial decryption adds an exponential error, there is an efficient distinguisher based on the evaluation error, and the scheme is not secure. We refer the reader to [LMSS22] for more details. The main technicality lies in the fact that, outputting a guess based on which of the two shifted Gaussians in more likely to have generated the challenge sample essentially provides an efficient distinguisher whose advantage is the statistical distance between the Gaussians, and an exponential flooding is required to make this statistical distance negligible.

As a consequence, it seems that relying on sanitization techniques to remove dependencies between the error term and the plaintexts involved in the computation is needed to obtain efficient threshold constructions without exponential noise flooding during partial decryption. This is what is done in [BS23a], by relying on circuit-private FHE. However, adding randomness before partial decryption rises another question: which party adds this randomness? If this is a party, then it has to be honest, uncorrupted, and even in this case, it is not clear that any security can be guaranteed with respect to this party, since it knows the randomness which serves as source of security. This is what motivates our generalization of threshold FHE, defined in the next section (Section 4): we add a $\mathsf{ServerDec}$ operation which processes the evaluated ciphertext to add randomness; we assume it is run by a trusted third-party (e.g., a server), so that none of the (possibly corrupted) parties knows the internal randomness of this step. Before we move on to our definition, let us analyze [BS23a] in more details.

3.2 A Construction Based on Circuit-Private FHE

In [BS23a], the authors proposed a transformation which allows to convert a One-Way-CPA (OW-CPA), circuit-private, threshold FHE into an IND-CPA threshold FHE. The precise definition of OW-CPA threshold FHE is not relevant for our work and we refer the reader to [BS23a] for more details. IND-CPA threshold FHE is similar to our Definition C.1, though it suffers from minor caveats. Indeed, in their security notions, the adversary does not have access to two distinct oracles OEval and ODec but only to a single oracle $\mathsf{OEvalDec}$ which combines evaluation of a circuit C with partial decryption of the resulting ciphertext. While this could be fine, the issue with their definition is that this oracle only reveals the partial decryption shares to the adversary and does not reveal the evaluated ciphertext which was decrypted. Note that, since the ciphertext is obtained by a circuit-private evaluation, it is not possible for the adversary to compute the decrypted ciphertext by itself. In practice, parties (and therefore the adversary) must know the ciphertext to be able to decrypt it, so $\mathsf{OEvalDec}$ should reveal both the ciphertext and the partial decryption shares when Eval is randomized. Yet, the fact that the ciphertext distribution is correlated to that of partial decryptions makes it much harder to analyze security. (Also, again, this rises the question of who does the evaluation since randomness plays a central role in security.)

Beside this definitional issue, our main contribution in this section is an attack against the transform. We start by briefly recalling the transform. Let $\mathsf{ThFHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$ be a OW-CPA, circuit-private, threshold FHE scheme. The authors suggest to construct an IND-CPA threshold FHE from this OW-CPA threshold FHE scheme by tweaking encryption and evaluation as follows.

- To encrypt a message $\mu \in \{0,1\}^\delta$: sample $(s_1, \ldots, s_\delta) \leftarrow U(\mathcal{M}^\delta)$ and $x \leftarrow U(\mathcal{M})$ where $\mathcal{M}$ denote the message space of the OW-CPA scheme – we assume that x has large min-entropy (i.e., that $\mathcal{M}$ is large); compute $\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x)$ and $r_i := \langle x, s_i \rangle \oplus \mu_i$ for $i \in [\delta]$; return $(\mathsf{ct}, (s_i)_{i\in\delta}, (r_i)_{i\in\delta})$.
- To evaluate a circuit C, one first unmasks the r_i's homomorphically using the encryption of x (and known s_i's) to compute an FHE encryption of μ, and then performs Eval of C on the resulting ciphertext.

Let us first remark that unmasking can be done directly on ciphertexts, and therefore one can assume that a message μ is simply encrypted using the OW-CPA scheme (up to some changes in the noise distribution). Therefore, the above transform would imply that a OW-CPA (circuit-private) threshold FHE scheme is actually IND-CPA-secure without any change. In Appendix C.4, we show that this is actually correct for *non-threshold* FHE schemes, assuming the scheme satisfies a mild form of circuit-privacy. The proof relies on a rewinding argument. However, if the attacker can make decryption queries (e.g., for threshold or IND-CPA-D security), the rewinding argument blows up the number of decryption queries and its seems very challenging to extend our result without this blow-up.[6]

3.3 An Attack Against the [BS23a] Transform

To attack the transform, we construct a threshold circuit-private FHE scheme which is does not satisfy their claimed security notion once plugged into their transform.

Syntax of our scheme. We consider a (threshold) FHE scheme with the following syntax. This corresponds in particular to some instantiations of B/FV or CKKS. The scheme uses a chain of moduli $q_\ell > q_{\ell-1} > \cdots > q_0$, providing $\ell+1$ levels of computation. Define $\mathcal{R} = \mathbb{Z}[X]/(X^N+1))$ for N a power of 2, and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ for $q \geq 2$. A ciphertext at level i is of the form $\mathsf{ct} = (a, -as+\mu+e)$ with $a, s, e \in \mathcal{R}_{q_i}$, where $\mu \in \mathcal{R}$ is a polynomial encoding a plaintext.

Let $\mathsf{ct}_0, \mathsf{ct}_1$ denote two level-$(i+1)$ ciphertexts with $\mathsf{ct}_\beta^{(i+1)} = (a_\beta, -a_\beta s + \mu_\beta + e_\beta)$ for $\beta \in \{0,1\}$. We assume that multiplying two level-$(i+1)$ ciphertexts leads to a level-i ciphertext $ct^{(i)}$ whose error term is of the form $\frac{q_i}{q_{i+1}} \cdot (\mu_0 e_1 + \mu_1 e_0 + e')$ where e' satisfies $\|e'\| \ll \|\mu_0 e_1 + \mu_1 e_0\|$. In the case of B/FV and CKKS, decreasing e' is obtained by increasing the q_i's.

We consider a circuit-private threshold FHE scheme with the above format and consider three consecutive moduli $q_2 > q_1 > q_0$ of the modulus chain. We assume that:

- fresh/evaluated ciphertexts are at level 0 (i.e., encryption, challenge, and evaluation queries all return level-0 ciphertexts);
- the public parameters contain a level-2 encryption of 1, denoted $\mathsf{ct}_{\mathsf{pp}}^{(2)}$;
- the (circuit-private) evaluation of a circuit C has the following form: it starts by performing a *circuit-private* evaluation of C resulting in a level-2 ciphertext $\mathsf{ct}_{\mathsf{priv}}^{(2)}$; then, the following post-processing is performed:
  1. both level-2 ciphertexts $\mathsf{ct}_{\mathsf{priv}}^{(2)}$ and $\mathsf{ct}_{\mathsf{pp}}^{(2)}$ are rescaled to level 1 by applying the map $c \mapsto \lfloor \frac{q_1}{q_2} \cdot c \rceil$; this results in two level-1 ciphertexts denoted $\mathsf{ct}_{\mathsf{priv}}^{(1)}$ and $\mathsf{ct}_{\mathsf{pp}}^{(1)}$, respectively;
  2. the ciphertexts $\mathsf{ct}_{\mathsf{priv}}^{(1)}$ and $\mathsf{ct}_{\mathsf{pp}}^{(1)}$ are multiplied, resulting in a level-0 ciphertext $\mathsf{ct}_{\mathsf{res}}^{(0)}$, which is returned as the output of the evaluation.

[6] Recall that, the OW-CPA threshold FHE construction proposed in [BS23a] requires to add a noise term of magnitude $O(Q_D)$ during partial decryption, with Q_D an upper bound on the number of decryption. Hence, if the reduction blows up the number of decryption queries to argue threshold IND-CPA security, then the amount of flooding noise shall be increased accordingly.

- partial decryption of a ciphertext (a, b) with a partial key s_j returns $as_j + \mathfrak{d}$ with $\mathfrak{d}$ being a noise of amplitude $\approx Q_D$, where Q_D denote an upper bound on the number of decryption queries made by the adversary (as suggested by the analysis based on the Rényi divergence from [BS23a]).

We now add two comments on the evaluation procedure. As $\mathsf{ct}_{\mathsf{pp}}^{(1)}$ is an encryption of 1, the ciphertext $\mathsf{ct}_{\mathsf{res}}^{(0)}$ properly decrypts to the result of the computation. Further note that the evaluation process is circuit-private since the post-processing is circuit-independent and applies to a circuit-private evaluated ciphertext $\mathsf{ct}_{\mathsf{priv}}^{(2)}$.

Finally, we assume that $q_0 \leq \mathsf{poly}(\lambda \cdot N \cdot Q_D)$, which suffices to enable correct decryption.

The attack. Our threshold IND-CPA attack starts as follows:

1. The attacker makes a challenge query (μ_0, μ_1) using scalar plaintexts. Let $\mathsf{ct}^* = (a^*, -a^*s + \mu_\beta + e^*)$ denote the resulting level-0 challenge ciphertext.
2. The attacker requests an encryption of the most significant bits of a^* (i.e., of a polynomial whose coefficients are the MSB of that of a^*; let us denote its encoding as $[a^*]$), resulting in a level-i ciphertext $\mathsf{ct}_1^{(0)}$.
3. The attacker requests an evaluation of the identity circuit on $\mathsf{ct}_1^{(0)}$, resulting in a level-0 ciphertext $\mathsf{ct}_2^{(0)}$.
4. The attacker finally requests all partial decryptions of $\mathsf{ct}_2^{(0)}$.

Note that the decryption query is valid as the underlying plaintext, which corresponds to the MSB of the a^*-part of ct^*, is independent of μ_β and therefore the decryption query does not reveal information about β.

Before completing the attack, let us analyze the various noise terms. By definition, $\mathsf{ct}_1^{(0)}$ is a level-0 ciphertext. When the identity circuit is evaluated, the circuit-private evaluation of the circuit is first run, ending up with a level-2 ciphertext $\mathsf{ct}_2^{(2)}$ of the form $(a_2^{(2)}, -a_2^{(2)}s + [a^*] + e_2^{(2)})$, which is then rescaled to a level-1 ciphertext $\mathsf{ct}_2^{(1)}$. Finally, it is multiplied to the rescaling $\mathsf{ct}_{\mathsf{pp}}^{(1)}$ of $\mathsf{ct}_{\mathsf{pp}}^{(2)}$. We have, for $k \in \{2, 1\}$:

$$\mathsf{ct}_{\mathsf{pp}}^{(k)} = (a_{\mathsf{pp}}^{(k)}, b_{\mathsf{pp}}^{(k)}) \quad \text{with} \quad a_{\mathsf{pp}}^{(k)}s + b_{\mathsf{pp}}^{(k)} = [1] + e_{\mathsf{pp}}^{(k)} \enspace ,$$

where [1] denotes the polynomial encoding of plaintext 1, and

$$\mathsf{ct}_2^{(k)} = (a_2^{(k)}, b_2^{(k)}) \quad \text{with} \quad a_2^{(k)}s + b_2^{(k)} = [a^*] + e_2^{(k)} \enspace .$$

Recall that $a_2^{(1)} := \lfloor \frac{q_1}{q_2} \cdot a_2^{(2)} \rceil$. Let r_2 denote the rounding error $r_2 := \frac{q_1}{q_2} a_2^{(2)} - a_2^{(1)}$. Then, the error term $e_2^{(1)}$ is $\lfloor r_2 s + \frac{q_1}{q_2} e_2^{(2)} \rceil$. By increasing q_2/q_1, we can make the error term $e_2^{(1)}$ dominated by $r_2 s$.

Similarly, letting r_{pp} denote the rounding error $r_{\mathsf{pp}} := \frac{q_1}{q_2} a_{\mathsf{pp}}^{(2)} - a_{\mathsf{pp}}^{(1)}$, we see that the error term $e_{\mathsf{pp}}^{(1)}$ is dominated by $r_{\mathsf{pp}} s$. Note that r_{pp} is publicly computable since $a_{\mathsf{pp}}^{(2)}$ is part of the public parameters and the rescaling operation is deterministic.

Finally, by definition of the scheme, multiplying $\mathsf{ct}_{\mathsf{pp}}^{(1)}$ with $\mathsf{ct}_2^{(1)}$ leads to an encryption $\mathsf{ct}_2 = (a_2, b_2)$ of $[a^*]$ with $b_2 = -a_2 s + [a^*] + e_2$ where the error term e_2 is of the form $[a^*]r_{\mathsf{pp}}s + r_2 s + e'$, where e' is small compared to the other terms. This error is then approximately $[a^*]r_{\mathsf{pp}}s$.

To complete the attack, we observe that the decryption of ct_2 allows the attacker to learn $a_2 s_j + \mathfrak{d}_j$ for $j \in [N]$, with N being the number of parties. Write $\mathfrak{d} := \sum_{j \in [N]} \mathfrak{d}_j$. Then, summing all partial decryptions with b_2, the adversary can recover $[a^*] + e_{\mathsf{dec}}$ where $e_{\mathsf{dec}} = \frac{q_i}{q_{i+1}} \cdot ([a^*]r_{\mathsf{pp}}s + [1]r_2 s + e') + \mathfrak{d}$, where $\|e'\| \ll \|[a^*]r_{\mathsf{pp}}s + [1]r_2 s\|$. Since the attacker knows $[a^*]$, it can recover e_{dec}.

The attacker knows r_{pp}, hence it can compute $r_{\mathsf{pp}}\mathsf{ct}^* = (a^* r_{\mathsf{pp}}, a^* s r_{\mathsf{pp}} + \mu_\beta r_{\mathsf{pp}} + e^* r_{\mathsf{pp}})$ and subtract a scaling of e_{dec} to the right-hand term. This has the effect of making the term $a^* s r_{\mathsf{pp}}$ small, as $[a^*]$ corresponds to the MSB of a^*. We then obtain a polynomial $\mu_\beta r_{\mathsf{pp}} + e_{\mathsf{small}}$, where $\|e_{\mathsf{small}}\| \ll q_0$. As the error term e_{small} is small compared to q_0, which is itself $\leq \mathsf{poly}(\lambda \cdot N \cdot Q)$, the error e_{small} is not sufficiently large to hide $\mu_\beta r_{\mathsf{pp}}$. By setting $\mu_0 = 0$ and $\mu_1 = [1]$, the attacker can guess β with non-negligible advantage based on the largeness of $\mu_\beta r_{\mathsf{pp}} + e_{\mathsf{small}}$.

Error in the analysis of the [BS23a] transform. We can trace our attack back to an error in the analysis of [BS23a, Theorem 3]: the issue lies in how [BS23a, Lemma 2] is applied in the proof. We borrow the notations from [BS23a, Lemma 2]. The set Z may depend on X, the first input of the extractor, but it cannot depend on its second input (the uniform distribution over $\{0,1\}^{n\delta}$). This implies that Z should not depend on the Goldreich-Levin bits s_j's in the proof of [BS23a, Theorem 3]. However, in that proof, it could depend on them: the s_j's are revealed in clear as part of the ciphertexts so the adversary could submit an encryption (or an evaluation) query whose plaintext (or circuit) depends on prior s_j's.

4 A Generalized Definition of Threshold FHE

In this section, we provide a generalization of the definition of threshold fully homomorphic encryption which allows to introduce (uncorrupted) randomness to a ciphertext before it is fed to partial decryption. We focus on N-out-of-N threshold FHE for readability and since our main construction (Section 5) handles only this setting. However, our definition extends to arbitrary monotone access structures. As already discussed, we consider that a trusted third-party (e.g., a server performing the computation) performs a public-key randomized pre-processing operation on ciphertexts before they can be partially decrypted. Specifically, the server applies an algorithm termed ServerDec, transforming a ciphertext into a form that is adequate for partial decryption by users. Using this terminology, evaluation could be deterministic and ServerDec randomized. We remark that, if ServerDec is void (or is deterministic), we recover the prior definition of threshold FHE.

We consider a simulation-based security notion inspired by [BGG+18]. Specifically, assume that the attacker corrupts parties $P_1, \ldots, P_{N-1}$ (and therefore knows $\mathsf{sk}_1, \ldots, \mathsf{sk}_{N-1}$). Let ct denote a ciphertext to be decrypted. It is first processed through $\mathsf{ServerDec}$. The result of this operation is a ciphertext $\mathsf{ct}_{\mathsf{dec}}$ which can be partially decrypted by all parties. Our security notion requires that $\mathsf{ct}_{\mathsf{dec}}$ and the partial decryptions of $\mathsf{ct}_{\mathsf{dec}}$ do not reveal more information than the plaintext μ underlying $\mathsf{ct}_{\mathsf{dec}}$ and what is already known by the adversary. This is enforced by the existence of a simulation which outputs a simulated $\mathsf{ct}_{\mathsf{dec}}$ and partial decryptions whose distribution is indistinguishable from that of the real ones. The sources of security are then clearly identified: (1) the internal randomness of $\mathsf{ServerDec}$ and (2) the randomness of the uncorrupted party P_N (i.e., the partial secret key sk_N and the internal randomness $\mathfrak{d}_N$ used when running $\mathsf{PartDec}$).

We now formalize our generalized definition of threshold functional encryption with the additional $\mathsf{ServerDec}$ algorithm.

Definition 4.1 (Threshold FHE). *A threshold fully homomorphic encryption scheme is a tuple of PPT algorithms* $\mathsf{ThFHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{ServerDec}, \mathsf{PartDec}, \mathsf{FinDec})$ *with the following properties. Let $\mathcal{M}$ denote the plaintext space, $\mathcal{C}$ the ciphertext space, $\mathcal{C}_{\mathsf{dec}}$ the processed ciphertext space, and $\mathcal{M}_{\mathsf{share}}$ the space of partial decryption shares.*

- $\mathsf{KeyGen}(1^\lambda, 1^N)$ *takes as input a security parameter, a number of parties N, and returns public parameters* pp *containing descriptions of $\mathcal{M}, \mathcal{C}, \mathcal{C}_{\mathsf{dec}}$, a public key* pk*, an evaluation key* ek*, and N secret key shares* $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$*;*
- $\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, \mu)$ *takes as input public parameters* pp*, a public key* pk*, and a plaintext $\mu \in \mathcal{M}$ and returns a ciphertext $\mathsf{ct} \in \mathcal{C}$;*
- $\mathsf{Eval}(\mathsf{pp}, \mathsf{ek}, \mathsf{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ *takes as input public parameters* pp*, an evaluation key* ek*, a circuit $\mathsf{C} : \mathcal{M}^\ell \to \mathcal{M}$ of arbitrary arity $\ell \geq 0$, and ℓ ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$, and returns a ciphertext $\mathsf{ct} \in \mathcal{C}$;*
- $\mathsf{ServerDec}(\mathsf{pp}, \mathsf{pk}, \mathsf{ct})$ *takes as input public parameters* pp*, a public key* pk*, and a ciphertext $\mathsf{ct} \in \mathcal{C}$ and returns a ciphertext $\mathsf{ct}_{\mathsf{dec}} \in \mathcal{C}_{\mathsf{dec}}$;*
- $\mathsf{PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct}_{\mathsf{dec}})$ *takes as input public parameters* pp*, a partial decryption key sk_i, and a ciphertext $\mathsf{ct}_{\mathsf{dec}} \in \mathcal{C}_{\mathsf{dec}}$, and returns a partial decryption $\mathsf{p}_i \in \mathcal{M}_{\mathsf{share}}$.*
- $\mathsf{FinDec}(\mathsf{pp}, \{\mathsf{p}_i\}_{i \in [N]})$ *takes as input public parameters* pp *and a set of partial decryptions $\{\mathsf{p}_i\}_{i \in [N]}$ in $\mathcal{M}_{\mathsf{share}}$, and returns a plaintext $\mu' \in \mathcal{M} \cup \{\perp\}$.*

To ease notation, the public parameters pp *are implicit for the rest of the paper. We require the following properties.*
Correctness. *For any $\lambda, N > 0$, $\ell \geq 0$, $\mathsf{C} : \mathcal{M}^\ell \to \mathcal{M}$, and $(\mu_1, \ldots, \mu_\ell) \in \mathcal{M}^\ell$, we have:*

$$\Pr\left[\mathsf{FinDec}(\{\mathsf{p}_i\}_{i \in [N]}) = y \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{ek}, (\mathsf{sk}_i)_{i \in [N]}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^N) \\ \mathsf{ct}_j \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu_j), \forall j \in [\ell] \\ \mathsf{ct} \leftarrow \mathsf{Eval}(\mathsf{ek}, \mathsf{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \\ \mathsf{ct}_{\mathsf{dec}} \leftarrow \mathsf{ServerDec}(\mathsf{pk}, \mathsf{ct}) \\ \mathsf{p}_i \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}_{\mathsf{dec}}), \forall i \in [N] \end{array}\right] \geq 1 - \mathsf{negl}(\lambda) \; ,$$

where $y = \mathsf{C}(\mu_1, \ldots, \mu_\ell)$ and the probability is over the internal coins of the algorithms.
Compactness. *There exists a polynomial p such that, for any $\lambda, N > 0$, and execution of $\mathsf{KeyGen}(1^\lambda, 1^N)$, we have:*

$$\log |\mathcal{C}|, \log |\mathcal{C}_{\mathsf{dec}}|, \log |\mathcal{M}_{\mathsf{share}}| \leq p(\lambda, N) \enspace .$$

Definition 4.2 (Threshold Simulation Security). *A threshold FHE scheme $\mathsf{ThFHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{ServerDec}, \mathsf{PartDec}, \mathsf{FinDec})$ is simulation-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 Sim such that the experiments depicted in Figure 1 are computationally indistinguishable; specifically, we require that for any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$,*

  $$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SimThFHE}} := \left| \Pr[\mathcal{A}(\mathsf{Exp}_{\mathsf{real}}^{\mathsf{ThFHE}}(1^\lambda, 1^N)) = 1] - \Pr[\mathcal{A}(\mathsf{Exp}_{\mathsf{ideal}}^{\mathsf{ThFHE}}(1^\lambda, 1^N)) = 1] \right|$$

  is negligible.

Our definition fixes a minor issue in [BGG+18, Definition 5.5]. In the latter definition, a state st is returned by Sim_1 when simulating KeyGen and fed as input to Sim_2. With this syntax, Sim_1 could run the actual KeyGen algorithm and st could contain all shares $(\mathsf{sk}_1, \ldots, \mathsf{sk}_N)$ of the secret key sk. In this case, Sim_2 can simply run the real evaluation/decryption algorithms, and simulation is perfect, but vacuous. Therefore, this definition could be trivially satisfied. In our definition, the public and partial keys revealed to the adversary are sampled identically in both experiments, using the KeyGen algorithm. It is very similar to the original definition from [JRS17] which did not suffer from the above minor issue, but our definition is slightly more general as it is multi-hop, adaptive (and we only require computational indistinguishability). It does not seem obvious to us that multi-hop, adaptive security is implied by single-hop, selective security since decryption results of multi-hop evaluation queries can be correlated.

Threshold public-key encryption. We also define a Threshold PKE scheme $\mathsf{ThPKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{PartDec}, \mathsf{FinDec})$ based on our definition of Threshold FHE by requiring Eval and $\mathsf{ServerDec}$ to be vacuous algorithms. A definition of simulation-based security is also obtained from simplifying the above definition by removing the OEval oracle and the $\mathsf{ServerDec}$ step in the oracle ODec (Step 4 in Figure 1).

5 Double-Flood-and-Round Construction

In this section, we propose an N-out-of-N threshold FHE scheme with small partial decryption shares. The design of our scheme is fairly generic and can be adapted to most FHE schemes for exact computations. We start by specifying some high-level structure for the underlying scheme.

```
$\mathsf{Exp}^{\mathsf{ThFHE}}_{\mathsf{real}}(1^\lambda, 1^N)$:
1: $(\mathsf{pk}, \mathsf{ek}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^N)$
2: $\mathsf{ctr} \leftarrow 0, \mathsf{L} \leftarrow \emptyset$
3: $(\mathcal{S}, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pk})$ with $|\mathcal{S}| = N-1$
4: $\mathsf{out} \leftarrow \mathcal{A}_1^{\mathsf{OEnc},\mathsf{OEval},\mathsf{ODec}}(\mathsf{pk}, (\mathsf{sk}_i)_{i\in\mathcal{S}}, \mathsf{st})$
5: Return $(\mathsf{out} = \mathsf{real})$
```

```
$\mathsf{Exp}^{\mathsf{ThFHE}}_{\mathsf{ideal}}(1^\lambda, 1^N)$:
1: $(\mathsf{pk}, \mathsf{ek}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^N)$
2: $\mathsf{ctr} \leftarrow 0, \mathsf{L} \leftarrow \emptyset$
3: $(\mathcal{S}, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pk})$ with $|\mathcal{S}| = N-1$
4: $\mathsf{out} \leftarrow \mathcal{A}_1^{\mathsf{OEnc},\mathsf{OEval},\mathsf{OSim}}(\mathsf{pk}, (\mathsf{sk}_i)_{i\in\mathcal{S}}, \mathsf{st})$
5: Return $(\mathsf{out} = \mathsf{ideal})$
```

```
$\mathsf{OEnc}(\mu)$:
1: $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu)$
2: $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
3: $\mathsf{L}[\mathsf{ctr}] \leftarrow (\mu, \mathsf{ct})$
4: Return $\mathsf{ct}$
```

```
$\mathsf{ODec}(j)$:
1: If $j > \mathsf{ctr}$:
2:     Return $\perp$
3: $(\mu, \mathsf{ct}) \leftarrow \mathsf{L}[j]$
4: $\mathsf{ct}_{\mathsf{dec}} \leftarrow \mathsf{ServerDec}(\mathsf{pk}, \mathsf{ct})$
5: $\mathsf{p}_k \leftarrow \mathsf{PartDec}(\mathsf{sk}_k, \mathsf{ct}_{\mathsf{dec}})$, for $k \in [N]$
6: Return $(\mathsf{ct}_{\mathsf{dec}}, (\mathsf{p}_k)_{k\in[N]})$
```

```
$\mathsf{OEval}(\mathsf{C}, (i_1, \ldots, i_\ell)))$:
1: For $j \in [\ell]$:
2:     $(\mu_j, \mathsf{ct}_j) \leftarrow \mathsf{L}[i_j]$
3: $\mathsf{ct} \leftarrow \mathsf{Eval}(\mathsf{ek}, \mathsf{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$
4: $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
5: $\mu \leftarrow \mathsf{C}(\mu_1, \ldots, \mu_\ell)$
6: $\mathsf{L}[\mathsf{ctr}] \leftarrow (\mu, \mathsf{ct})$
7: Return $\mathsf{ct}$
```

```
$\mathsf{OSim}(j)$:
1: If $j > \mathsf{ctr}$:
2:     Return $\perp$
3: $(\mu, \mathsf{ct}) \leftarrow \mathsf{L}[j]$
4: $(\mathsf{ct}_{\mathsf{dec}}, (\mathsf{p}_k)_{k\in[N]}) \leftarrow \mathsf{Sim}(\mathsf{pk}, \mu, (\mathsf{sk}_i)_{i\in\mathcal{S}})$
5: Return $(\mathsf{ct}_{\mathsf{dec}}, (\mathsf{p}_k)_{k\in[N]})$
```

Fig. 1. Simulation security games for Threshold FHE.

5.1 Structure of the Underlying FHE Scheme

Let λ denote a security parameter. Let $\mathsf{FHE} = (\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$ denote an FHE scheme with the following structure.

1. The public parameters include two dimensions $m \geq n \geq 1$, a modulus $Q \geq 2$, a secret key distribution $\chi_{\mathbf{s}}$ over $\mathbb{Z}^n$ whose samples have norms $\leq \mathsf{poly}(\lambda)$, and three distributions χ_e, χ_v and χ_f over $\mathbb{Z}$. All these are functions of λ, and the distributions are assumed to be efficiently sampleable.
2. The key pair is of the form:[7]

$$\mathsf{pk} = [\mathbf{A}|\mathbf{b}] \text{ with } \mathbf{b} := -\mathbf{A}\mathbf{s} + \mathbf{e} \ , \ \mathsf{sk} = \mathbf{s} \ ,$$

with $\mathbf{A} \leftarrow U(\mathbb{Z}_Q^{m\times n})$, $\mathbf{s} \leftarrow \chi_{\mathbf{s}}$ and $\mathbf{e} \leftarrow \chi_e^m$. We have $\mathsf{pk} \cdot (\mathsf{sk}, 1)^\intercal = \mathbf{e} \bmod Q$.

[7] We ignore the evaluation key ek in our description and our proof. This is common even for non-threshold schemes: security of FHE relies on the additional assumption that security still holds given ek (which typically involves a circular security assumption).

3. Encrypting a plaintext $\mu \in \mathcal{M}$ starts by encoding μ into some plaintext $\mu' = \mathsf{Encode}_q(\mu) \in \mathbb{Z}_Q$ (e.g., by setting $\mu' = \Delta \cdot \mu$ for some scaling factor Δ). Then encryption first computes $\mathsf{ct}_0, \mathsf{ct}_1$ with:
$$\mathsf{ct}_0 := \mathbf{v}^\mathsf{T}\mathbf{A} + \mathbf{f}^\mathsf{T} \ , \ \ \mathsf{ct}_1 := \mathbf{v}^\mathsf{T}\mathbf{b} + f' + \mu' \ ,$$
with $\mathbf{v} \leftarrow \chi_v^m$, $\mathbf{f} \leftarrow \chi_f^n$ and $f' \leftarrow \chi_f$. Note that this setting handles the case where both $\mathbf{f}$ and f' are zero (for instance in FHE schemes where the leftover hash lemma is relied upon for IND-CPA). The ciphertext is $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$.
4. The decryption algorithm Dec is split into two steps $(\mathsf{Dec}_1, \mathsf{Dec}_2)$, as follows. Let ct be its input ciphertext (which can be either a fresh ciphertext or the result of an homomorphic computation).
   (a) $\mathsf{Dec}_1(\mathsf{sk}, \mathsf{ct})$ returns $\mathsf{z} := \mathsf{ct}_0 \cdot \mathsf{sk} + \mathsf{ct}_1 \bmod Q$.
   (b) $\mathsf{Dec}_2(\mathsf{z})$ returns $\mu := \mathsf{Decode}_Q(\mathsf{z})$. Importantly, it does not use sk; it might be void, notably in the case of an LWE version of CKKS, whereas it typically involves a rounding or a modular reduction in exact schemes.

   We assume that $\mathsf{z} = \mu' + e_{\mathsf{eval}}$ where $\mu' = \mathsf{Encode}_Q(\mu)$ is an encoding of the plaintext μ corresponding to ct and e_{eval} is an error term with bounded magnitude $|e_{\mathsf{eval}}| \leq B_{\mathsf{eval}}$ for some $B_{\mathsf{eval}} > 0$. We refer to e_{eval} as the evaluation error.

Our framework captures most known (LWE-based) FHE schemes. It does not directly capture GSW, in which messages are encoded as $\mu \cdot \mathbf{G}$ for $\mathbf{G}$ the gadget matrix and encrypted as $\mathsf{ct} = \mathbf{R} \cdot \mathsf{pk} + \mathbf{F} + \mu \cdot \mathbf{G}$. We could generalize our description to also encompass GSW but it would hurt the readability regarding the decryption procedure, hence we chose this less general but simpler description.

5.2 Construction

As already mentioned, a simple solution to obtain a threshold FHE (even for general access structure or when the computation is performed by parties) is to have the parties add an exponential noise term after decryption such that no information about their partial decryption key is revealed to other. Yet, a significant drawback of the noise flooding approach is the size of the output ciphertexts, as the modulus needs to be large enough to tolerate the addition of this large noise term at decryption. To mitigate this, we propose a different approach, in which exponential noise flooding is performed on the server side. Computation by the server is performed with a large modulus Q which tolerates exponential noise-flooding, and the ciphertext is then rounded to a smaller modulus $q_{\mathsf{dec}} \ll Q$, before being sent to the users. This modulus remains sufficiently large for the users to be able to add some limited noise term to guarantee security without impacting decryption correctness. After performing its partial decryption, a user can round the decryption share to an even smaller modulus: indeed, there is no more noise that needs to be added, and it is only required that the combination of the current noises does not impact correctness. Overall, the successive roundings allow to minimize bandwidth consumption.[8]

The scheme is described in Figures 2 and 3. To ease description, our scheme does not include the second rounding (performed after partial decryption). We briefly discuss this optimization at the end of this section. The construction relies on a perfectly correct FHE scheme that fulfills the constraints of Section 5.1. For the sake of simplicity, we consider an FHE scheme that encodes a plaintext μ over $\mathbb{Z}_Q$ as $\mathsf{Encode}_Q(\mu)$ such that for $Q = pq_{\mathsf{dec}}$, it holds that: $1/p \cdot \mathsf{Encode}_Q(\mu) = \mathsf{Encode}_{q_{\mathsf{dec}}}(\mu)$. For example, this holds if plaintexts are encoded in the most significant bits of the ciphertexts, i.e., $\mathsf{Encode}_Q(\mu) = Q/t \cdot \mu$, with t denoting the plaintext modulus (e.g., as done in B/FV).

We only describe the procedures $\mathsf{KeyGen}, \mathsf{ServerDec}, \mathsf{PartDec}, \mathsf{FinDec}$ in Figures 2 and 3, as the encryption and evaluation procedures Enc and Eval are identical to those of the underlying FHE scheme (and operate over $\mathbb{Z}_Q$ with Q being the larger modulus). The scheme involves two noise flooding parameters σ_{flood} and η. The first flooding parameter σ_{flood} is used for exponential flooding, while the second parameter η is used for small flooding during partial decryption. It also involves two moduli $Q = pq_{\mathsf{dec}}$ and q_{dec} with $q_{\mathsf{dec}} \ll Q$ (we use $p \approx 2^{\lambda}$). $\mathsf{ServerDec}$ uses randomized Gaussian roundings. We reveal the norm of the secret key $\|\mathsf{sk}\|$ in the public parameters. This is only to ease simulation in our security analysis. In practice, the scheme is at least as secure if $\|\mathsf{sk}\|$ is not given, since removing it only restricts the information available to an attacker.

```
▶ $\mathsf{KeyGen}(1^{\lambda}, 1^{N})$:
  1: Construct $Q = p \cdot q_{\mathsf{dec}}$ with $q_{\mathsf{dec}} \ll Q$ and the secret key dimension $n$
  2: Sample $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{FHE.KeyGen}(1^{\lambda})$ with modulus $Q$
  3: Define public parameters $\mathsf{pp}$ containing $Q, p, q_{\mathsf{dec}}$ as well as $\|\mathsf{sk}\|$
  4: Sample $(\mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Share}(\mathsf{sk}, N, q_{\mathsf{dec}})$
  5: Return $(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, \mathsf{pk}, \mathsf{pp})$
```

Fig. 2. Key generation of double-flood-and-round threshold FHE.

The communication involved between parties after the computation is limited, as only the small modulus q_{dec} is involved. However, communication before the computation, to provide inputs to the server, remains large as we encrypt over $\mathbb{Z}_Q$. This may be solved using transciphering (see, e.g., [BCK+23] and references therein).

The construction can be adapted to further reduce the bandwidth. The users could apply a second rounding step after their partial decryption, e.g., by returning $\lfloor q_{\mathsf{out}}/q_{\mathsf{dec}} \cdot \mathsf{p}_i \rceil \bmod q_{\mathsf{out}}$ for $q_{\mathsf{out}} < q_{\mathsf{dec}}$, in order to further reduce the

[8] Another drawback of exponential flooding, which our construction does not address, is the need for an LWE parametrization with exponential noise rate.

```
▶ ServerDec(pk, ct):
  1: $\mathsf{ct}_{\mathsf{fresh}} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(0)$
  2: $\mathsf{ct}_{\mathsf{in}} \leftarrow \mathsf{ct} + \mathsf{ct}_{\mathsf{fresh}}$
  3: $\mathfrak{E} \leftarrow \mathcal{D}_{\sigma_{\mathsf{flood}}}$
  4: Parse $\mathsf{ct}_{\mathsf{in}}$ as $(\mathsf{ct}_{\mathsf{in},0}, \mathsf{ct}_{\mathsf{in},1}) \bmod Q$
  5: $\mathsf{ct}_{\mathsf{in},1} \leftarrow \mathsf{ct}_{\mathsf{in},1} + \mathfrak{E}$
  6: $\mathsf{ct}_{\mathsf{dec},0} := \left\lfloor \frac{1}{p} \cdot \mathsf{ct}_{\mathsf{in},0} \right\rceil_{\sigma_0} \bmod q_{\mathsf{dec}}$
  7: $\mathsf{ct}_{\mathsf{dec},1} := \left\lfloor \frac{1}{p} \cdot \mathsf{ct}_{\mathsf{in},1} \right\rceil_{\sigma_1} \bmod q_{\mathsf{dec}}$
  8: $\mathsf{ct}_{\mathsf{dec}} := (\mathsf{ct}_{\mathsf{dec},0}, \mathsf{ct}_{\mathsf{dec},1})$
  9: Return $\mathsf{ct}_{\mathsf{dec}}$

▶ PartDec($\mathsf{sk}_i, \mathsf{ct}_{\mathsf{dec}}$):
  1: Parse $\mathsf{ct}_{\mathsf{dec}}$ as $(\mathsf{ct}_{\mathsf{dec},0}, \mathsf{ct}_{\mathsf{dec},1})$
  2: Sample $\mathfrak{d}_i \leftarrow \mathcal{D}_{\mathbb{Z},\eta}$
  3: $\mathsf{p}_i \leftarrow \mathsf{ct}_{\mathsf{dec},0} \cdot \mathsf{sk}_i + \mathfrak{d}_i \bmod q_{\mathsf{dec}}$
  4: Return $\mathsf{p}_i$

▶ FinDec($\mathsf{ct}_{\mathsf{dec}}, \{\mathsf{p}_i\}_{i \in [N]}$):
  1: Parse $\mathsf{ct}_{\mathsf{dec}}$ as $(\mathsf{ct}_{\mathsf{dec},0}, \mathsf{ct}_{\mathsf{dec},1})$
  2: $\mathsf{z} \leftarrow \sum_{i \in [N]} \mathsf{p}_i + \mathsf{ct}_{\mathsf{dec},1} \bmod q_{\mathsf{dec}}$
  3: $\mu \leftarrow \mathsf{Decode}_{q_{\mathsf{dec}}}(\mathsf{z})$
  4: Return $\mu$
```

Fig. 3. Decryption procedures of double-flood-and-round threshold FHE.

size of communication with other parties. Security follows from that of the base construction since the modified scheme provides strictly less information to the adversary. Functionality is preserved as long as parameters are carefully selected to ensure correctness.

5.3 Analysis of the Double-Flood-and-Round Construction

Let B_{eval} be an upper bound on $|\mathsf{ct} \cdot \mathsf{sk} - \mathsf{Encode}_Q(\mu)|$ for any ciphertext ct that can be produced by a combination of encryptions and evaluations, and where μ is the underlying plaintext of ct. We assume that fresh ciphertexts also have decryption noises that are bounded by B_{eval} in absolute value (this follows from the definition of B_{eval} if FHE.Eval does not do anything for the empty circuit).

Theorem 5.1. *Let* ThFHE *denote the above double-flood-and-round construction. It is a correct and secure threshold fully homomorphic encryption scheme, assuming that:*

- *the decoding procedure of the underlying FHE scheme satisfies*

$$\mathsf{Decode}_{q_{\mathsf{dec}}}\left(\mathsf{Encode}_{q_{\mathsf{dec}}}(\mu) + e\right) = \mu \ ,$$

  for any plaintext μ and e with $|e| \leq \sqrt{\lambda} \cdot \sigma_{\mathsf{dec}}$ and where σ_{dec} is defined as $\sqrt{(\sigma_0\|\mathbf{s}\|)^2 + (\sigma_{\mathsf{flood}}/p)^2 + \sigma_1^2 + N\eta^2}$;
- *the underlying FHE scheme is* IND-CPA*-secure;*

- $\sigma_0, \sigma_1 = \Omega(\sqrt{\lambda + \log n})$ *and* $\sigma_{\mathsf{flood}} = \Omega(p\|\mathsf{sk}\|\sqrt{\lambda+\log n})$*;*
- $\sigma_{\mathsf{flood}} = \Omega(2^{\lambda} B_{\mathsf{eval}})$*;*
- $\mathsf{yaLWE}_{n,Q_D,q_{\mathsf{dec}},\sigma_{\mathfrak{e}},\eta,\chi_{\mathsf{s}}}$ *holds for* $\sigma_{\mathfrak{e}} = \sqrt{(\sigma_0\|\mathbf{s}\|)^2 + (\sigma_{\mathsf{flood}}/p)^2 + \sigma_1^2}$*;*
- $\mathsf{LWE}_{n,Q,\chi_f,\chi_{\mathsf{s}}}$ *holds.*

In the above, the variable Q_D *refers to the number of decryption queries made by the adversary. We assume that* $Q_D \leq \mathsf{poly}(\lambda)$*.*

Proof. We prove correctness and threshold simulation security independently. IND-CPA security follows from that of the underlying FHE scheme.

Correctness. Let $\mathsf{ct}_{\mathsf{dec}} = (\mathsf{ct}_{\mathsf{dec},0}, \mathsf{ct}_{\mathsf{dec},1})$ denote a ciphertext obtained from the server after it applied $\mathsf{ServerDec}$. We define the rounding error $\mathbf{r}_0$ of $\mathsf{ct}_{\mathsf{dec},0}$ as:

$$\mathbf{r}_0^{\mathsf{T}} := \mathsf{ct}_{\mathsf{in},0} - p \cdot \mathsf{ct}_{\mathsf{dec},0} \ .$$

By definition, since $\mathsf{ct}_{\mathsf{dec},0} \sim \mathcal{D}_{\mathbb{Z}^n, \frac{1}{p}\cdot\mathsf{ct}_{\mathsf{in},0},\sigma_0}$, we have $\mathbf{r}_0 \sim \mathcal{D}_{p\{\frac{1}{p}\cdot\mathsf{ct}_{\mathsf{in},0}\}+p\mathbb{Z}^n, p\sigma_0}$, where $\{\cdot\}$ denotes the fractional part defined as $\{x\} := x - \lfloor x \rceil$ for any $x \in \mathbb{R}$. Assume that $\mathsf{ct}_{\mathsf{in},1} = \mathsf{ct}_{\mathsf{in},0} \cdot \mathsf{sk} + \mu' + e_{\mathsf{eval}} + e_{\mathsf{fresh}}$, where e_{eval} is the decryption noise of the input ct of $\mathsf{ServerDec}$ and e_{fresh} is the decryption noise of $\mathsf{ct}_{\mathsf{fresh}}$ (recall that $\mathsf{ct}_{\mathsf{in}} = \mathsf{ct} + \mathsf{ct}_{\mathsf{fresh}}$). We then have, modulo q_{dec}:

$$\begin{aligned}\mathsf{ct}_{\mathsf{dec},1} &= \left\lfloor \frac{1}{p} \cdot (-(p \cdot \mathsf{ct}_{\mathsf{dec},0} + \mathbf{r}_0^{\mathsf{T}}) \cdot \mathsf{sk} + \mu' + e_{\mathsf{eval}} + e_{\mathsf{fresh}} + \mathfrak{E}) \right\rceil_{\sigma_1} \\ &= -\mathsf{ct}_{\mathsf{dec},0} \cdot \mathsf{sk} + \left\lfloor \frac{1}{p} \cdot (-\mathbf{r}_0^{\mathsf{T}} \cdot \mathsf{sk} + \mu' + e_{\mathsf{eval}} + e_{\mathsf{fresh}} + \mathfrak{E}) \right\rceil_{\sigma_1} \\ &= \mathsf{Encode}_{q_{\mathsf{dec}}}(\mu) - \mathsf{ct}_{\mathsf{dec},0} \cdot \mathsf{sk} + \left\lfloor \frac{1}{p} \cdot (-\mathbf{r}_0^{\mathsf{T}} \cdot \mathsf{sk} + e_{\mathsf{eval}} + e_{\mathsf{fresh}} + \mathfrak{E}) \right\rceil_{\sigma_1} \ . \quad (1)\end{aligned}$$

Let $\mathsf{p}_i = \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}_{\mathsf{dec}})$, and recall that $\mathsf{p}_i = \mathsf{ct}_{\mathsf{dec},0} \cdot \mathsf{sk}_i + \mathfrak{d}_i$, with $\mathfrak{d}_i \leftarrow \mathcal{D}_{\mathbb{Z},\eta}$ for $i \in [N]$. Then, we have, modulo q_{dec}:

$$\begin{aligned}\mathsf{FinDec}((\mathsf{p}_i)_{i\in[N]}) &= \left(\sum_{i\in[N]} \mathsf{p}_i\right) + \mathsf{ct}_{\mathsf{dec},1} \\ &= \mathsf{ct}_{\mathsf{dec},0} \cdot \mathsf{sk} + \mathsf{ct}_{\mathsf{dec},1} + \sum_{i\in[N]} \mathfrak{d}_i \\ &= \mathsf{Encode}_{q_{\mathsf{dec}}}(\mu) + \mathfrak{e} + \sum_{i\in[N]} \mathfrak{d}_i \ , \quad (2)\end{aligned}$$

where $\mathfrak{e} := \left\lfloor \frac{1}{p} \cdot (-\mathbf{r}_0^{\mathsf{T}} \cdot \mathsf{sk} + e_{\mathsf{eval}} + e_{\mathsf{fresh}} + \mathfrak{E}) \right\rceil_{\sigma_1}$. We finally obtain:

$$\mathsf{FinDec}((\mathsf{p}_i)_{i\in[N]}) = \mathsf{Encode}_{q_{\mathsf{dec}}}(\mu) + e_{\mathsf{dec}} \bmod q_{\mathsf{dec}} \ ,$$

with $e_{\mathsf{dec}} = \mathfrak{e} + \sum_{i\in[N]} \mathfrak{d}_i$. Correctness follows as long as $|e_{\mathsf{dec}}|$ is sufficiently small to enable correct decoding to μ. In our security analysis below, we show in the proof of Lemma 5.3 that e_{dec} follows a distribution which is statistically close to $\mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{dec}}}$, where $\sigma_{\mathsf{dec}} = \sqrt{(\sigma_0\|\mathbf{s}\|)^2 + (\sigma_{\mathsf{flood}}/p)^2 + \sigma_1^2 + N\eta^2}$. By standard Gaussian tail bounds, the samples have magnitudes $\leq \sqrt{\lambda}\sigma_{\mathsf{dec}}$ with overwhelming probability.

Threshold simulation. As for all known examples of LWE-based FHE scheme, we ignore the evaluation key in our security analysis. Our actual security claim is then obtained by additionally assuming that security still holds provided the extra information contained in the evaluation key. This is the standard circular security assumption underlying FHE schemes.

Our proof proceeds by a sequence of hybrids. The $\mathsf{KeyGen}, \mathsf{Enc}$ and Eval algorithms are modified in none of them and encryption and evaluation queries are answered by running the corresponding algorithms. Hence, we only focus on handle decryption queries.

Let $\mathcal{A}$ denote an adversary. Without loss of generality, assume that $\mathcal{A}$ corrupts parties $1, \ldots, N-1$ so that it knows $\mathsf{sk}_1, \ldots, \mathsf{sk}_{N-1}$. Our objective is to prove that the real experiment distribution $\mathsf{Exp}_{\mathsf{real}}^{\mathsf{ThFHE}}$ is computationally indistinguishable from the simulated one $\mathsf{Exp}_{\mathsf{ideal}}^{\mathsf{ThFHE}}$, which can be run directly given the information of $\mathcal{A}$ (i.e., given the corrupted partial keys and the plaintexts underlying the ciphertexts that are queried to the decryption oracle, but without sk nor sk_N). The simulator Sim is given in Figure 5.3.

```
▶ $\mathsf{Sim}(\mathsf{pk}, \mu, (\mathsf{sk}_1, \ldots, \mathsf{sk}_{N-1}))$:
1: $\mathsf{ct}_{\mathsf{dec},0} \leftarrow U(\mathbb{Z}_{q_{\mathsf{dec}}}^n)$
2: $\mathsf{ct}_{\mathsf{dec},1} \leftarrow U(\mathbb{Z}_{q_{\mathsf{dec}}})$
3: $\mathfrak{d}_i \leftarrow \mathcal{D}_{\mathbb{Z},\eta}$ for $i \in [N-1]$
4: $\mathsf{p}_i \leftarrow \mathsf{ct}_{\mathsf{dec},0} \cdot \mathsf{sk}_i + \mathfrak{d}_i$ for $i \in [N-1]$
5: $\mathfrak{h} \leftarrow \mathcal{D}_{\sigma_{\mathfrak{h}}}$ where $\sigma_{\mathfrak{h}} := \sqrt{(\sigma_0\|\mathbf{s}\|)^2 + (\sigma_{\mathsf{flood}}/p)^2 + \sigma_1^2 + \eta^2}$
6: $\mathsf{p}_N \leftarrow \mathsf{Encode}_{q_{\mathsf{dec}}}(\mu) + \mathfrak{h} - \mathsf{ct}_{\mathsf{dec},1} - \sum_{i\in[N-1]} \mathsf{ct}_{\mathsf{dec},0} \cdot \mathsf{sk}_i$
7: Return $(\mathsf{ct}_{\mathsf{dec},0}, \mathsf{ct}_{\mathsf{dec},1}, \mathsf{p}_1, \ldots, \mathsf{p}_N)$.
```

Fig. 4. Simulator for the double-flooding-and-round threshold FHE.

Let ct denote a ciphertext held by the server and for which the adversary is requesting decryption. Let μ denote the underlying plaintext. Note that ct could be a fresh encryption of μ or the result of a homomorphic computation whose underlying plaintext is μ. We aim to prove that $\mathsf{ct}_{\mathsf{dec}} \leftarrow \mathsf{ServerDec}(\mathsf{pk}, \mathsf{ct})$, the ciphertext revealed by the server to all parties, and $\mathsf{p}_N \leftarrow \mathsf{PartDec}(\mathsf{sk}_N, \mathsf{ct}_{\mathsf{dec}})$, the partial decryption of the uncorrupted party N, are computationally indistinguishable from those provided by the simulator Sim. We proceed by a hybrid argument, first considering the real distribution in $\mathsf{Exp}_{\mathsf{real}}^{\mathsf{ThFHE}}$.

Hyb$_0$. This is the adversary's view in $\mathsf{Exp}_{\mathsf{real}}^{\mathsf{ThFHE}}$. Given the constraints we imposed on the underlying FHE scheme, the ciphertext $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$ to be decrypted satisfies:

$$\mathsf{ct}_0 \cdot \mathsf{sk} + \mathsf{ct}_1 = \mu' + e_{\mathsf{eval}} \bmod Q \ ,$$

with $\mu' = \mathsf{Encode}_Q(\mu)$ and $|e_{\mathsf{eval}}| \leq B_{\mathsf{eval}}$.

$\mathbf{Hyb}_1$. In this first hybrid, we change how $\mathsf{ct}_{\mathsf{in}}$ is defined by the challenger. We remind that, as for ct, the ciphertext $\mathsf{ct}_{\mathsf{in}}$ is held by the server and is never revealed to the adversary. The latter only sees $\mathsf{ct}_{\mathsf{dec}}$, which is produced from $\mathsf{ct}_{\mathsf{in}}$.

In this hybrid, when the adversary makes a decryption query for a ciphertext ct encrypting a plaintext μ', the server now samples $(\mathsf{ct}_{\mathsf{in},0}, \mathsf{ct}_{\mathsf{in},1})$ and $\mathsf{ct}_{\mathsf{dec}}$ as follows:

- it samples $\mathsf{ct}_{\mathsf{in},0} \leftarrow \mathbf{a}^{\intercal}$ with $\mathbf{a} \leftarrow U(\mathbb{Z}_Q^n)$;
- it sets $\mathsf{ct}_{\mathsf{in},1} \leftarrow -\mathbf{a}^{\intercal} \cdot \mathsf{sk} + \mu' + \mathfrak{E}$, where $\mathfrak{E} \leftarrow \mathcal{D}_{\sigma_{\mathsf{flood}}}$;
- the rest of the decryption proceeds as before.

We claim that games $\mathbf{Hyb}_0$ and $\mathbf{Hyb}_1$ are computationally indistinguishable. The detailed analysis is provided in Lemma 5.2.

We recall the correctness equation (Equation (2)):

$$\left(\sum_{i\in[N]} \mathsf{p}_i\right) + \mathsf{ct}_{\mathsf{dec},1} = \mathsf{Encode}_{q_{\mathsf{dec}}}(\mu) + \mathfrak{e} + \sum_{i\in[N]} \mathfrak{d}_i \bmod q_{\mathsf{dec}} \ , \tag{3}$$

where $\mathfrak{e}$ is now $\mathfrak{e} := \left\lfloor \frac{1}{p} \cdot (-\mathbf{r}_0^{\intercal} \cdot \mathsf{sk} + \mathfrak{E}) \right\rceil_{\sigma_1}$, with $\mathbf{r}_0 \sim \mathcal{D}_{p\{\frac{1}{p}\cdot \mathsf{ct}_{\mathsf{in},0}\}+p\mathbb{Z}^n, p\sigma_0}$.

$\mathbf{Hyb}_2$. In this hybrid, we change how the partial decryption p_N of the uncorrupted party is computed as well as how $\mathsf{ct}_{\mathsf{dec},1}$ is sampled. Simplifying Equation (3) above, we obtain that, in $\mathbf{Hyb}_1$, the partial decryption p_N satisfies:

$$\mathsf{p}_N := \mathsf{Encode}_{q_{\mathsf{dec}}}(\mu) + \mathfrak{e} + \mathfrak{d}_N - \mathsf{ct}_{\mathsf{dec},1} - \sum_{i\in[N-1]} \mathsf{ct}_{\mathsf{dec},0} \cdot \mathsf{sk}_i \ . \tag{4}$$

Motivated by this equation, in $\mathbf{Hyb}_2$, we now sample $\mathsf{ct}_{\mathsf{dec},1}$ as $\mathsf{ct}_{\mathsf{dec},1} \leftarrow U(\mathbb{Z}_{q_{\mathsf{dec}}})$ and set p_N as:

$$\mathsf{p}_N := \mathsf{Encode}_{q_{\mathsf{dec}}}(\mu) + \mathfrak{h} - \mathsf{ct}_{\mathsf{dec},1} - \sum_{i\in[N-1]} \mathsf{ct}_{\mathsf{dec},0} \cdot \mathsf{sk}_i \ ,$$

with $\mathfrak{h} \leftarrow \mathcal{D}_{\sigma_{\mathfrak{h}}}$ and $\sigma_{\mathfrak{h}} := \sqrt{(\sigma_0\|\mathbf{s}\|)^2 + (\sigma_{\mathsf{flood}}/p)^2 + \sigma_1^2 + \eta^2}$.

We claim that games $\mathbf{Hyb}_1$ and $\mathbf{Hyb}_2$ are computationally indistinguishable. The detailed analysis is provided in Lemma 5.3. Note that the distribution in this game no longer depends on sk_N and is therefore sampleable by the adversary given its known information, since $\sigma_0, \sigma_1, p, \sigma_{\mathsf{flood}}, \|\mathbf{s}\|, \eta$ are public parameters. The view is identical to the one provided by the Sim algorithm from Figure 5.3.

This completes the proof of Theorem 5.1. □

Lemma 5.2. *Assuming that $\sigma_{\mathsf{flood}} = \Omega(2^{\lambda} B_{\mathsf{eval}})$ and that $\mathsf{LWE}_{n,Q,\chi_f,\chi_{\mathsf{s}}}$ holds, games $\mathbf{Hyb}_0$ and $\mathbf{Hyb}_1$ are computationally indistinguishable.*

Proof. For simplicity, we focus on simulating a single decryption query: the general case where multiple decryption queries are made is obtained by a standard hybrid argument.

Let $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$ denote a ciphertext corresponding to a decryption query made by the adversary. First, we recall that a fresh encryption $\mathsf{ct}_{\mathsf{fresh}}$ of 0 is added to the ciphertext ct at Step 2 of $\mathsf{ServerDec}(\mathsf{ct})$ to produce $\mathsf{ct}_{\mathsf{in}}$. It is of the form $\mathsf{ct}_{\mathsf{fresh}} = (\mathsf{ct}_{\mathsf{fresh},0}, \mathsf{ct}_{\mathsf{fresh},1})$ with $\mathsf{ct}_{\mathsf{fresh},0} = \mathbf{v}^\intercal \mathbf{A} + \mathbf{f}^\intercal$ and $\mathsf{ct}_{\mathsf{fresh},1} = -\mathsf{ct}_{\mathsf{fresh},0} \cdot \mathbf{s} + e_{\mathsf{fresh}}$. Here $\mathbf{v} \leftarrow \chi_v^m$, $\mathbf{f} \leftarrow \chi_f^n$ and the noise term e_{fresh} satisfies $|e_{\mathsf{fresh}}| \leq B_{\mathsf{eval}}$. Further, given our assumptions regarding the underlying FHE scheme, the ciphertext $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$ to be decrypted satisfies:

$$\mathsf{ct}_0 \cdot \mathbf{s} + \mathsf{ct}_1 = \mu' + e_{\mathsf{eval}} \bmod Q \ ,$$

with $\mu' = \mathsf{Encode}_Q(\mu)$ and $|e_{\mathsf{eval}}| \leq B_{\mathsf{eval}}$. Overall, the ciphertext $\mathsf{ct}_{\mathsf{in}}$ is of the form $(\mathsf{ct}_{\mathsf{in},0}, \mathsf{ct}_{\mathsf{in},1})$ with:

$$\mathsf{ct}_{\mathsf{in},0} = \mathbf{v}^\intercal \mathbf{A} + \mathbf{f}^\intercal + \mathsf{ct}_0$$

and

$$\mathsf{ct}_{\mathsf{in},1} = -\mathsf{ct}_{\mathsf{in},0} \cdot \mathbf{s} + e_{\mathsf{fresh}} + e_{\mathsf{eval}} + \mathfrak{E} + \mu' \ .$$

Note that $|e_{\mathsf{fresh}} + e_{\mathsf{eval}}| \leq 2B_{\mathsf{eval}}$. By Lemma 2.1, taking $\sigma_{\mathsf{flood}} = \Omega(2^\lambda B_{\mathsf{eval}})$, the above distribution of $\mathsf{ct}_{\mathsf{in}}$ is statistically indistinguishable from sampling $\mathsf{ct}_{\mathsf{in}}$ as:

$$\mathsf{ct}_{\mathsf{in},0} = \mathbf{v}^\intercal \mathbf{A} + \mathbf{f}^\intercal + \mathsf{ct}_0 \quad , \quad \mathsf{ct}_{\mathsf{in},1} \leftarrow -\mathsf{ct}_{\mathsf{in},0} \cdot \mathbf{s} + \mathfrak{E} + \mu' \ .$$

Finally, note that $\mathsf{ct}_{\mathsf{in},1}$ no longer contains information about $\mathbf{v}, \mathbf{f}$ apart from that carried by $\mathsf{ct}_{\mathsf{in},0}$. In the above, we can hence replace $\mathbf{v}^\intercal \mathbf{A} + \mathbf{f}^\intercal$ by a uniformly random value over $\mathbb{Z}_Q^n$, under the LWE assumption. As a result, the distribution of $\mathsf{ct}_{\mathsf{in}}$ is computationally indistinguishable from a pair of the form $(\mathbf{a}^\intercal, -\mathbf{a}^\intercal \mathbf{s} + \mathfrak{E} + \mu')$ where $\mathbf{a} \leftarrow U(\mathbb{Z}_Q^n)$, which is precisely the distribution of $\mathsf{ct}_{\mathsf{in}}$ in $\mathbf{Hyb}_1$. □

Lemma 5.3. *Assuming that* $\mathsf{yaLWE}_{n,Q_D,q_{\mathsf{dec}},\sigma_{\mathfrak{e}},\eta,\chi_{\mathsf{s}}}$ *holds, games* $\mathbf{Hyb}_1$ *and* $\mathbf{Hyb}_2$ *are computationally indistinguishable.*

Proof. We aim to prove that the view of the adversary in games $\mathbf{Hyb}_1$ and $\mathbf{Hyb}_2$ are computationally indistinguishable. In both games, the ciphertext $\mathsf{ct}_{\mathsf{in}}$ is defined as $(\mathsf{ct}_{\mathsf{in},0}, \mathsf{ct}_{\mathsf{in},1}) \leftarrow (\mathbf{a}^\intercal, \mathbf{a}^\intercal \mathsf{sk} + \mu' + \mathfrak{E})$. Then the vector $\mathsf{ct}_{\mathsf{dec},0}$ is computed as

$$\mathsf{ct}_{\mathsf{dec},0} \leftarrow \left\lfloor \frac{1}{p} \cdot \mathbf{a}^\intercal \right\rceil_{\sigma_0} \bmod q_{\mathsf{dec}} \ ,$$

which is revealed to the adversary. Note that, since $\mathbf{a} \sim U(\mathbb{Z}_Q^n)$ and p divides Q, since only $\mathsf{ct}_{\mathsf{dec}}$ is revealed to the adversary, one can directly sample $\mathsf{ct}_{\mathsf{dec},0}$ uniformly over $\mathbb{Z}_{q_{\mathsf{dec}}}^n$. Using the same notation as before (in the proof of correctness), we define the rounding error of $\mathsf{ct}_{\mathsf{dec},0}$ as $\mathbf{r}_0^\intercal := \mathsf{ct}_{\mathsf{in},0} - p \cdot \mathsf{ct}_{\mathsf{dec},0}$, and recall that, by definition, we have $\mathbf{r}_0 \sim \mathcal{D}_{p\{\frac{1}{p} \cdot \mathsf{ct}_{\mathsf{in},0}\} + p\mathbb{Z}^n, p\sigma_0}$.

The adversary's view in $\mathbf{Hyb}_1$ is then $(\mathsf{ct}_{\mathsf{dec},0}, \mathsf{ct}_{\mathsf{dec},1}, (\mathsf{p}_i)_{i \in [N]})$ where $\mathsf{ct}_{\mathsf{dec},0}$ is defined as above, and $(\mathsf{p}_i)_{i \in [N-1]}$ can be computed directly by the adversary since it knows $\mathsf{sk}_1, \ldots, \mathsf{sk}_{N-1}$ and $\mathsf{ct}_{\mathsf{dec},0}$. It remains to deal with $\mathsf{ct}_{\mathsf{dec},1}$ and p_N. By adapting Equation (1) to $\mathbf{Hyb}_1$, we have, modulo q_{dec}:

$$\mathsf{ct}_{\mathsf{dec},1} = \mathsf{Encode}_{q_{\mathsf{dec}}}(\mu) - \mathsf{ct}_{\mathsf{dec},0} \cdot \mathsf{sk} + \left\lfloor \frac{1}{p} \cdot (-\mathbf{r}_0^{\mathsf{T}} \cdot \mathsf{sk} + \mathfrak{E}) \right\rceil_{\sigma_1} .$$

Further, from Equation (4), we have, modulo q_{dec}:

$$\mathsf{p}_N = \mathsf{Encode}_{q_{\mathsf{dec}}}(\mu) + \left\lfloor \frac{1}{p} \cdot (-\mathbf{r}_0^{\mathsf{T}} \cdot \mathsf{sk} + \mathfrak{E}) \right\rceil_{\sigma_1} + \mathfrak{d}_N - \mathsf{ct}_{\mathsf{dec},1} - \sum_{i=\in[N-1]} \mathsf{ct}_{\mathsf{dec},0} \cdot \mathsf{sk}_i .$$

Since μ, $\mathsf{ct}_{\mathsf{dec},0}$ and $(\mathsf{sk}_i)_{i \in [N-1]}$ are known to the adversary, and replacing $\mathsf{ct}_{\mathsf{dec},1}$ in the second equation the right hand side of the first equation, we observe that it suffices to focus on the quantities (defined modulo q_{dec}):

$$\mathsf{ct}'_{\mathsf{dec},1} := -\mathsf{ct}_{\mathsf{dec},0} \cdot \mathsf{sk} + \left\lfloor \frac{1}{p} \cdot (-\mathbf{r}_0^{\mathsf{T}} \cdot \mathsf{sk} + \mathfrak{E}) \right\rceil_{\sigma_1} \quad \text{and} \quad \mathsf{p}'_N := -\mathsf{ct}_{\mathsf{dec},0} \cdot \mathsf{sk} - \mathfrak{d}_N .$$

The partial decryption noise $\mathfrak{d}_N$ having a distribution that is symmetric aroung 0, up to inverting the sign, letting $\mathbf{a} := -\mathsf{ct}_{\mathsf{dec},0}$ and $\mathbf{s} := \mathsf{sk}$, we then have:

$$\mathsf{ct}'_{\mathsf{dec},1} = \mathbf{a} \cdot \mathbf{s} + \mathfrak{e} \bmod q_{\mathsf{dec}} \quad \text{and} \quad \mathsf{p}'_N = \mathbf{a} \cdot \mathbf{s} + \mathfrak{d}_N \bmod q_{\mathsf{dec}} ,$$

where $\mathfrak{e} := \left\lfloor \frac{1}{p} \cdot (-\mathbf{r}_0^{\mathsf{T}} \cdot \mathsf{sk} + \mathfrak{E}) \right\rceil_{\sigma_1}$.

Recall that $(1/p) \cdot \mathbf{r}_0^{\mathsf{T}}$ follows the distribution $\mathcal{D}_{\{\frac{1}{p} \cdot \mathsf{ct}_{\mathsf{in},0}\} + \mathbb{Z}^n, \sigma_0}$ and $\mathfrak{E}/p$ follows the distribution $\mathcal{D}_{\frac{1}{p}\sigma_{\mathsf{flood}}}$. Therefore, applying Lemma 2.2, assuming $(1/\sigma_0^2 + (p\|\mathsf{sk}\|/\sigma_{\mathsf{flood}})^2)^{-1/2} \geq \eta_\varepsilon(\mathbb{Z}^n)$ for some $\varepsilon < 1/2$, the distribution of $-(1/p) \cdot \mathbf{r}_0^{\mathsf{T}} \cdot \mathsf{sk} + \mathfrak{E}/p$ is at statistical distance at most 4ε from $\mathcal{D}_{\sqrt{(\sigma_0\|\mathbf{s}\|)^2 + (\sigma_{\mathsf{flood}}/p)^2}}$. The smoothing condition is fulfilled, thanks to the assumptions on σ_0 and σ_{flood}. By definition of Gaussian rounding and thanks to the latter observation, we can then apply Lemma 2.3. Assuming $\sigma_1 \geq \eta_\varepsilon(\mathbb{Z})$ for some $\varepsilon < 1/2$, we then obtain that the distribution of $\mathfrak{e}$ is within statistical distance 8ε of the discrete Gaussian $\mathcal{D}_{\mathbb{Z}, \sqrt{(\sigma_0\|\mathbf{s}\|)^2 + (\sigma_{\mathsf{flood}}/p)^2 + \sigma_1^2}}$. The smoothing condition is fulfilled, thanks to the assumption on σ_1.

Define $\sigma_{\mathfrak{e}} := \sqrt{(\sigma_0\|\mathbf{s}\|)^2 + (\sigma_{\mathsf{flood}}/p)^2 + \sigma_1^2}$. Now that we have proven that $\mathfrak{e}$ is statistically close from $\mathcal{D}_{\mathbb{Z},\sigma_{\mathfrak{e}}}$, the rest of the proof is similar to the simulation proof for threshold PKE from [MS23], since $\mathsf{ct}_{\mathsf{dec}}$ is essentially a fresh PKE ciphertext now. Recall that $\mathfrak{d}_N \sim \mathcal{D}_{\mathbb{Z},\eta}$. The pair $(\mathsf{ct}'_{\mathsf{dec},1}, \mathsf{p}'_N)$ precisely corresponds to a sample for the yaLWE problem, for secret $\mathbf{s}$. Thanks to the privacy of Share, the adversary has no information about $\mathbf{s}$ (except the knowledge of $\|\mathbf{s}\|$ which is publicly available), even given $\mathsf{sk}_1, \ldots, \mathsf{sk}_{N-1}$, since the latter are identically distributed as $\mathsf{sk}'_1, \ldots, \mathsf{sk}'_{N-1}$ where $(\mathsf{sk}'_1, \ldots, \mathsf{sk}'_N) \leftarrow \mathsf{Share}(\mathbf{0}, N, q_{\mathsf{dec}})$, which are independent of $\mathbf{s}$. Assuming that $\mathsf{yaLWE}_{n, q_{\mathsf{dec}}, \sigma_{\mathfrak{e}}, \eta, \chi_{\mathbf{s}}}$ holds, we obtain that $(\mathsf{ct}'_{\mathsf{dec},1}, \mathsf{p}'_N)$ from $\mathbf{Hyb}_1$ is computationally indistinguishable from a pair $(\mathsf{ct}'_{\mathsf{dec},1}, \mathsf{p}'_N)$ sampled as:

$$\mathsf{ct}_{\mathsf{dec},1} \leftarrow U(\mathbb{Z}_{q_{\mathsf{dec}}}) \quad \text{and} \quad \mathsf{p}_N \leftarrow \mathsf{ct}_{\mathsf{dec},1} + \mathfrak{h} ,$$

with $\mathfrak{h} \leftarrow \mathcal{D}_{\sigma_\mathfrak{h}}$ where $\sigma_\mathfrak{h} := \sqrt{(\sigma_0\|\mathbf{s}\|)^2 + (\sigma_{\mathsf{flood}}/p)^2 + \sigma_1^2 + \eta^2}$.

Since $\sigma_0, \sigma_1, p, \sigma_{\mathsf{flood}}, \|\mathbf{s}\|$ and η are public parameters, the latter distribution is publicly sampleable and precisely corresponds to the distribution generated by our simulator, i.e., to the distribution in $\mathbf{Hyb}_2$ (up to the terms known to the adversary that we ignored above for simplicity).

We complete the proof of Lemma 5.3 by applying a hybrid argument on all ciphertexts for which a decryption query is made. □

5.4 Open Problems

The construction can be extended to a t-out-of-N threshold FHE, by relying on Shamir secret sharing or linear integer secret sharing [DT06]. However, we do not know how to adapt the security analysis, in particular how to simulate all $N - t + 1$ partial decryptions that are not available to the adversary. We now describe a way to partially circumvent this difficulty, by relying on N-out-of-N threshold FHE. For each each subset $\mathcal{S} \subseteq [N]$ of size t, compute a t-additive secret sharing of sk. This leads to $\binom{N}{t}$ independent additive secret sharings of sk. Then, the partial key of each party P_i is the union of shares of the key for each valid set $\mathcal{S}$ such that $i \in \mathcal{S}$. This induces a significative blow-up, but for small choices of t and N, the overhead is limited.

Similarly, while the scheme can be extended to the ring setting, extending the analysis to rely on ring-LWE [SSTX09,LPR10] seems challenging. Most of the proof extends to the ring setting, but there is one specific difficulty that arises: revealing $\|\mathbf{s}\|$ is sufficient to obtain a simulator in the LWE case, it does not seem to be no longer the case in the ring setting. Letting $s \in \mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ denote the secret key (with N a power of 2), directly extending the analysis would require to reveal the covariance $s\bar{s}$ where $\bar{s}$ denotes the polynomial $s(X^{-1})$. It may however be possible to extend the security analysis to the ring setting by relying on the extension of ring-LWE proposed in [MS23, Section 5.3] (in the context of threshold PKE).

References

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

ASY22. Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. In *ICALP*, 2022.

BCK$^+$23. Youngjin Bae, Jung Hee Cheon, Jaehyung Kim, Jai Hyun Park, and Damien Stehlé. HERMES: efficient ring packing using MLWE ciphertexts and application to transciphering. In *CRYPTO*, 2023.

BCKS24. Youngjin Bae, Jung Hee Cheon, Jaehyung Kim, and Damien Stehlé. Bootstrapping bits with CKKS. In *EUROCRYPT*, 2024.

BD10. Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In *TCC*, 2010.

BF11. Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In *PKC*, 2011.

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.

BI22. Florian Bourse and Malika Izabachène. Plug-and-play sanitization for TFHE. *IACR eprint 2022/1438*, 2022.

BJKL21. Fabrice Benhamouda, Aayush Jain, Ilan Komargodski, and Huijia Lin. Multiparty reusable non-interactive secure computation from LWE. In *EUROCRYPT*, 2021.

BKSS24. Youngjin Bae, Jaehyung Kim, Eias Suvanto, and Damien Stehlé. Bootstrapping small integers with CKKS. In *ASIACRYPT*, 2024.

BLL$^+$18. Shi Bai, Adeline Langlois, Tancrède Lepoint, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. *Journal of Cryptology*, 2018.

BLP$^+$13. Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, 2013.

BPMW16. Florian Bourse, Rafaël Del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In *CRYPTO*, 2016.

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

BS23a. Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus. In *ASIACRYPT*, 2023.

BS23b. Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus. *IACR eprint 2023/016*, 2023. Version dated 16 July 2024.

CCP$^+$24. Jung Hee Cheon, Hyeongmin Choe, Alain Passelègue, Damien Stehlé, and Elias Suvanto. Attacks against the IND-CPAD security of exact FHE schemes. In *CCS*, 2024.

CGGI16. Ilaria Chillotti, Nicolas Gama, Mariya 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 Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT*, 2017.

CSB$^+$24. Marina Checri, Renaud Sirdey, Aymen Boudguiga, Jean-Paul Bultel, and Antoine Choffrut. On the practical CPAD security of "exact" and threshold FHE schemes and libraries. In *IACR eprint 2024/116*, 2024.

CSS$^+$22. Siddhartha Chowdhury, Sayani Sinha, Animesh Singh, Shubham Mishra, Chandan Chaudhary, Sikhar Patranabis, Pratyay Mukherjee, Ayantika Chatterjee, and Debdeep Mukhopadhyay. Efficient threshold FHE with application to real-time systems. *IACR eprint 2022/1625*, 2022. Version dated 18 July 2024.

DDK$^+$23. Morten Dahl, Daniel Demmler, Sarah El Kazdadi, Arthur Meyre, Jean-Baptiste Orfila, Dragos Rotaru, Nigel P. Smart, Samuel Tap, and Michael Walter. Noah's ark: Efficient threshold-fhe using noise flooding. In *WAHC*, 2023.

DM15. Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In *EUROCRYPT*, 2015.

DMPS24. Nir Drucker, Guy Moshkowich, Tomer Pelleg, and Hayim Shaul. BLEACH: cleaning errors in discrete computations over CKKS. *Journal of Cryptology*, 2024.

DS16. Léo Ducas and Damien Stehlé. Sanitization of FHE ciphertexts. In *EUROCRYPT*, 2016.

DT06. Ivan Damgård and Rune Thorbek. Linear integer secret sharing and distributed exponentiation. In *PKC*, 2006.

DWF22. Xiaokang Dai, Wenyuan Wu, and Yong Feng. Key lifting : a more efficient weak MKFHE scheme in the plain model against rational adversary. *IACR eprint 2022/055*, 2022.

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

Gen09. Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, 2009.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. An efficient and parallel gaussian sampler for lattices. In *STOC*, 2008.

JRS17. Aayush Jain, Peter M. R. Rasmussen, and Amit Sahai. Threshold fully homomorphic encryption. In *eprint 2017/257*, 2017.

Klu22. Kamil Kluczniak. Circuit privacy for fhew/tfhe-style fully homomorphic encryption in practice. *IACR eprint 2022/1459*, 2022.

KS23. Kamil Kluczniak and Giacomo Santato. On circuit private, multikey and threshold approximate homomorphic encryption. In *IACR eprint 2023/301*, 2023.

LM21. Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In *EUROCRYPT*, 2021.

LMSS22. Baiyu Li, Daniele Micciancio, Mark Schultz, and Jessica Sorrell. Securing approximate homomorphic encryption using differential privacy. In *CRYPTO*, 2022.

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

MM11. Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In *CRYPTO*, 2011.

MS23. Daniele Micciancio and Adam Suhl. Simulation-secure threshold PKE from LWE with polynomial modulus. In *IACR eprint 2023/1728*, 2023.

Pei10. Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO*, 2010.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.

Shi22. Sina Shiehian. mrNISC from LWE with polynomial modulus. In *SCN*, 2022.

SSTX09. Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, 2009.

A Proof of Lemma 2.1

We believe tighter bounds exist in the literature but could not find one, so we provide a simple proof for the claim from Lemma 2.1. Let $\sigma > 0, c_0, c_1 \in \mathbb{Z}$.

Assume that $c_0 \geq c_1$. We have:

$$\begin{aligned}\mathit{\Delta}(\mathcal{D}_{\mathbb{Z},c_0,\sigma}, \mathcal{D}_{\mathbb{Z},c_1,\sigma}) &= \frac{1}{2\sigma} \sum_{x\in\mathbb{Z}} \left| \exp\left(\frac{-\pi(x-c_0)^2}{\sigma^2}\right) - \exp\left(\frac{-\pi(x-c_1)^2}{\sigma^2}\right) \right| \\ &\leq \frac{|c_0-c_1|}{2\sigma\rho_\sigma(\mathbb{Z})} \sum_{x\in\mathbb{Z}} \max_{x-c_0\leq t\leq x-c_1} \left| \frac{-2\pi t}{\sigma^2} \cdot \exp\left(\frac{-\pi t^2}{\sigma^2}\right)\right|\end{aligned}$$

via the mean value theorem. Let $\alpha = \pi/\sigma^2$. Since $t \mapsto -2\alpha t \cdot \exp\left(-\alpha t^2\right)$ is symmetric, non-decreasing except on $(\frac{-1}{\sqrt{2\alpha}}, \frac{1}{\sqrt{2\alpha}})$, and reaches its maximum in $\frac{-1}{\sqrt{2\alpha}}$ with the value $\sqrt{2\alpha}\exp(-1/2)$, we have:

$$\begin{aligned}\frac{2\sigma\rho_\sigma(\mathbb{Z})}{|c_0-c_1|} \cdot \mathit{\Delta}(\mathcal{D}_{\mathbb{Z},c_0,\sigma}, \mathcal{D}_{\mathbb{Z},c_1,\sigma}) &\leq \int_{-\infty}^{\frac{-1}{\sqrt{2\alpha}}} -2\alpha t \cdot \exp(-\alpha t^2) dt \\ &\quad + (\frac{2}{\sqrt{2\alpha}} + (c_0-c_1)) \cdot \sqrt{2\alpha}\exp\left(-1/2\right) \\ &\quad + \int_{\frac{1}{\sqrt{2\alpha}}}^{\infty} 2\alpha t \cdot \exp(-\alpha t^2) dt \\ &\leq 4\sqrt{\alpha} \int_0^\infty t\sqrt{\alpha} \cdot \exp\left((t\sqrt{\alpha})^2\right) dt + |c_0-c_1|\sqrt{2\alpha} \ ,\end{aligned}$$

where the last equation follows from $\int_0^\infty t\exp(-t^2)dt = 1/2$. Finally, replacing α by π/σ^2, we obtain:

$$\mathit{\Delta}(\mathcal{D}_{\mathbb{Z},c_0,\sigma}, \mathcal{D}_{\mathbb{Z},c_1,\sigma}) \leq \frac{2|c_0-c_1|}{2\sigma\rho_\sigma(\mathbb{Z})} + \frac{\sqrt{2\pi}|c_0-c_1|^2}{2\sigma^2\rho_\sigma(\mathbb{Z})} \ ,$$

Hence, fixing c_0, c_1, we have $\mathit{\Delta}(\mathcal{D}_{\mathbb{Z},c_0,\sigma}, \mathcal{D}_{\mathbb{Z},c_1,\sigma}) = O\left(\frac{|c_0-c_1|}{\sigma}\right)$ when $\sigma \to \infty$, which concludes the proof of Lemma 2.1. □

B A One-More Round Protocol Based on Threshold PKE

In this section, we describe a simpler but weaker protocol that relies on threshold PKE and circuit-private (non-threshold) FHE. The protocol is not round-optimal, as it requires one more round during which a party, called the *transcryptor*, enables the transition from FHE to threshold PKE. While circuit-privacy is easily obtained from (exponential) noise-flooding techniques, there are alternative constructions which allow to obtain more compact ciphertexts. This makes this protocol a potentially interesting alternative to our double-flood-and-round threshold FHE scheme. Note however that the protocol described in this section is not a threshold FHE scheme properly speaking: it allows to perform threshold computation in a private manner using a trusted server, but it requires an additional round for decryption (as well as secure channels between parties and the server performing the computation). Moreover, since we rely on threshold PKE in black-box, this protocol extends to arbitrary access structures (as long as the threshold PKE scheme supports them) and is not restricted to the N-out-of-N setting.

We first recall the definition of circuit-privacy for (non-threshold) FHE.

Definition B.1 (Circuit-Privacy). *An FHE scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *with message space* $\mathcal{M}$ *is said to be circuit-private if for any circuit* $\mathsf{C} : \mathcal{M}^\ell \to \mathcal{M}$*, we have:*

$$\varDelta((\mathsf{pk}, \mathsf{sk}, \mathsf{Eval}(\mathsf{ek}, \mathsf{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)), (\mathsf{pk}, \mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{C}(\mu_1, \ldots, \mu_\ell)))) \leq \mathsf{negl}(\lambda) \enspace ,$$

where $\mu_i \in \mathcal{M}$*,* $\mathsf{ct}_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mu_i)$ *for* $i \in [\ell]$*, and where* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*.*

The protocol considers 3 types of parties: the users, who are interested in performing a joint computation on their private data, a trusted server who performs the homomorphic computation, and the transcryptor which is either a semi-honest third-party or a user. It must participate in the protocol (in an N-out-of-N scenario, this could be any party, but our scenario extends to arbitrary access structures handled by the underlying threshold PKE, in which case some parties might not participate; the transcryptor must always participate). The protocol is as follows. For simplicity, we focus on the N-out-of-N case.

Let $\mathsf{ThPKE} = (\mathsf{ThKeyGen}, \mathsf{ThEnc}, \mathsf{ThPartDec}, \mathsf{ThFinDec})$ denote a threshold PKE scheme and $\mathsf{FHE} = (\mathsf{FhKeyGen}, \mathsf{FhEnc}, \mathsf{FhEval}, \mathsf{FhDec})$ be a circuit-private fully homomorphic encryption scheme. Consider N users who want to perform a threshold computation. These users share a threshold PKE secret key $\mathsf{tsk}_1, \ldots, \mathsf{tsk}_N$ corresponding to a threshold PKE public key tpk. A user participating in the decryption or a third-party is designated as the transcryptor: it generates a pair of FHE keys $(\mathsf{fsk}, \mathsf{fpk})$ and reveals fpk to all other parties (users and server). *We further assume that communication between parties and between a party and the server is done via a secure channel using end-to-end encryption, though we do not write this layer of encryption to ease the reading.*

Our protocol is described in Figure 5. To simplify the description, we ignore the layer of end-to-end encryption which is added on top of all communications.

Analysis of the protocol. We assume that the server and the transcryptor are non-colluding (together, they have access to all data and can decrypt everything). The server is trusted (it is semi-honest and uncorrupted).

The above protocol is not a threshold FHE scheme as per Definition 4.2, but it allows to perform private joint computation using a trusted server. We choose to keep this section slightly informal in order to avoid redefining formally the protocol syntax as well as the security definition. We claim that the protocol satisfies simulation security: any adversary which corrupts up to $N-1$ parties (and possibly the transcryptor) does not learn any viable information about the N-th uncorrupted party's secret data (its private input to the computation and its partial decryption key). This is modeled by the existence of a simulator as for threshold FHE. Specifically, assuming P_N is the uncorrupted party, there

- **Setup:**
  1. Parties $P_1, \ldots, P_N$ run the threshold PKE key generation protocol to compute $(\mathsf{tpk}, \mathsf{tsk}_1, \ldots, \mathsf{tsk}_N) \leftarrow \mathsf{ThKeyGen}(1^\lambda)$ so that P_i gets $(\mathsf{tpk}, \mathsf{tsk}_i)$ for $i \in [N]$;
  2. The transcryptor generates $(\mathsf{fpk}, \mathsf{fsk}) \leftarrow \mathsf{FhKeyGen}(1^\lambda)$ and broadcasts fpk to the server and all parties. It keeps fsk for itself.
- **Encrypt data to the server:**
  1. To send a data μ_i to the server, a user P_i computes $\mathsf{fhct}_i \leftarrow \mathsf{FhEnc}_{\mathsf{fpk}}(\mu_i)$ and sends it to the server.
- **Evaluation of C over encrypted data:** To compute a circuit C on data $\mu_1, \ldots, \mu_\ell$ encrypted by users, the server evaluates $\mathsf{ThEnc}_{\mathsf{tpk}} \circ \mathsf{C}$ homomorphically on ciphertexts fhct_i's received from the users, where $\mathsf{fhct}_i := \mathsf{FhEnc}_{\mathsf{fpk}}(\mu_i)$ for $i \in [\ell]$. The result is a ciphertext $\mathsf{fhct}_{\mathsf{res}} \leftarrow \mathsf{FhEval}(\mathsf{ThEnc}_{\mathsf{tpk}} \circ \mathsf{C}, \mathsf{fhct}_1, \ldots, \mathsf{fhct}_\ell)$ which decrypts to $\mathsf{ThEnc}_{\mathsf{tpk}}(\mathsf{C}(\mu_1, \ldots, \mu_\ell)$.
- **Decryption:**
  1. The server sends $\mathsf{fhct}_{\mathsf{res}}$ to the transcryptor;
  2. The transcryptor decrypts it using fsk to recover a TPKE ciphertext $\mathsf{tct}_{\mathsf{res}} = \mathsf{ThEnc}_{\mathsf{tpk}}(\mathsf{C}(\mu_1, \ldots, \mu_\ell)$;
  3. The transcryptor broadcasts $\mathsf{tct}_{\mathsf{res}}$ to all parties;
  4. Party P_i computes and broadcasts $\mathsf{p}_i \leftarrow \mathsf{ThPartDec}(\mathsf{sk}_i, \mathsf{tct}_{\mathsf{res}})$;
  5. Parties combine partial decryptions to recover the result as $\mathsf{ThFinDec}((\mathsf{p}_i)_{i \in [N]})$.

Fig. 5. A 3-round protocol for private threshold computation

exists an efficient simulator Sim, which takes as inputs the result $\mathsf{C}(\mu_1, \ldots, \mu_\ell)$ of the computation as well as the corrupted secret information $(\mathsf{tsk}_1, \ldots, \mathsf{tsk}_{N-1}$, possibly fsk, etc.) and returns a tuple $(\mathsf{fhct}_{\mathsf{res}}, \mathsf{tct}_{\mathsf{res}}, \mathsf{p}_N)$ whose distribution is computationally indistinguishable from that of $(\mathsf{fhct}_{\mathsf{res}}, \mathsf{tct}_{\mathsf{res}}, \mathsf{p}_N)$ in the honest execution of the protocol.

Theorem B.2. *Assuming* ThPKE *and* FHE *are correct, the above protocol is correct. Furthermore, assuming* ThPKE *is simulation-secure and that* FHE *is circuit-private, the above protocol is simulation-secure.*

Proof. We argue about correctness of the protocol, as well as privacy of the data with respect to the server, and privacy of the data with respect to corrupted parties.

Correctness of the protocol. Correctness of the protocol immediately follows from the correctness of the underlying FHE and ThPKE schemes.

Security of users data against the server. The server sees only FHE encryptions of the users data. If the users communicate using end-to-end encryption during the decryption protocol, the server does see anything else (and in particular, does not see $\mathsf{tct}_{\mathsf{res}}$, the decryption of the FHE ciphertext it computed), and then standard $\mathsf{IND\text{-}CPA}$ security of FHE (together with security of end-to-end encryption) guarantees that it does not learn anything about the data.

Security of users data against corrupted users. Let us now assume that an adversary corrupts $N-1$ parties. Wlog, let us assume it corrupts parties $P_1, \ldots, P_{N-1}$. Furthermore, let us assume that the adversary also corrupts the transcryptor (that is, either it is one of the $N-1$ corrupted parties, or it is a third-party that is also corrupted). We aim to guarantee the security of the data of the last uncorrupted party P_N. Suppose parties $P_1, \ldots, P_N$ engage a protocol to jointly compute $\mathsf{C}(\mu_1, \ldots, \mu_N)$ where μ_i is provided by party P_i, for $i \in [N]$. The adversary sees the following information which depend on P_N's private information (tsk_N or μ_N):

- The FHE ciphertext $\mathsf{fhct}_{\mathsf{res}}$ computed by the server and sent to the transcryptor;
- Its FHE decryption $\mathsf{tct}_{\mathsf{res}}$;
- The partial decryption p_N of $\mathsf{tct}_{\mathsf{res}}$ computed by P_N.
- Communication between party P_N and the server, which is protected by end-to-end encryption.

In addition, note that the adversary knows public parameters, as well as keys $\mathsf{tpk}, \mathsf{fpk}, \mathsf{fsk}, \mathsf{tsk}_1, \ldots, \mathsf{tsk}_{N-1}$ as well as $\mu_1, \ldots, \mu_{N-1}$. Our goal is to show that the above information $\mathsf{fhct}_{\mathsf{res}}, \mathsf{tct}_{\mathsf{res}}, \mathsf{p}_N$ and the private communication between P_N and the server can be simulated knowing only the result $\mathsf{C}(\mu_1, \ldots, \mu_N)$ of the computation.

We define a brief sequence of hybrid games. The communication between party P_N and the server being encrypted using end-to-end encryption, we can ignore it (it can be simulated as a communication with only 0's being encrypted thanks to $\mathsf{IND\text{-}CPA}$ security of the end-to-end encryption). We focus on the rest of the information available to the adversary, i.e. on the FHE ciphertext $\mathsf{fhct}_{\mathsf{res}}$, the TPKE ciphertext $\mathsf{tct}_{\mathsf{res}}$, and its partial decryption p_N by the uncorrupted party P_N.

$\mathbf{Hyb}_0$. This first distribution corresponds to the honest distribution in the above protocol.

$\mathbf{Hyb}_1$. In this first hybrid, instead of computing $\mathsf{fhct}_{\mathsf{res}}$ by evaluating homomorphically $\mathsf{ThEnc}_{\mathsf{pk}} \circ \mathsf{C}$, the challenger computes it by first computing $\mathsf{tct}_{\mathsf{res}} \leftarrow \mathsf{ThEnc}_{\mathsf{pk}}(\mathsf{C}(\mu_1, \ldots, \mu_N)$ (using the same randomness for ThEnc as in $\mathbf{Hyb}_0$) and then encrypting $\mathsf{fhct}_{\mathsf{res}} \leftarrow \mathsf{FhEnc}_{\mathsf{fpk}}(\mathsf{tct})$. Thanks to the circuit-privacy of FHE, the two hybrid games are statistically indistinguishable.

$\mathbf{Hyb}_2$. In this second hybrid, the challenger now relies on the threshold PKE simulator $\mathsf{Sim}_{\mathsf{ThPKE}}$ to simulate $\mathsf{tct}_{\mathsf{res}} = \mathsf{ThEnc}_{\mathsf{tpk}}(\mathsf{C}(\mu_1, \ldots, \mu_N))$ together with p_N. That is, it now first samples

$$(\mathsf{tct}_{\mathsf{res}}, \mathsf{p}_N) \leftarrow \mathsf{Sim}_{\mathsf{ThPKE}}(\mathsf{tpk}, \mathsf{tsk}_1, \ldots, \mathsf{tsk}_{N-1}, \mathsf{C}(\mu_1, \ldots, \mu_N)) \ ,$$

and then proceeds as before, except that it reveals the simulated partial decryption p_N instead of the honestly generated one. By definition of simulation security of ThPKE, $\mathbf{Hyb}_1$ and $\mathbf{Hyb}_2$ are computationally indistinguishable.

This concludes the analysis, since the distribution sampled by the challenger in $\mathbf{Hyb}_2$ is sampleable given the information known to the adversary. □

Remark B.3 (On the need for circuit-privacy). Above, we gave a security analysis assuming the transcryptor is corrupted. The case where the transcryptor is not corrupted is simpler since the adversary has strictly less information. Notably, it can be proven secure without assuming circuit-privacy: indeed, if the transcryptor is not corrupted, the adversary never learns $\mathsf{fhct}_{\mathsf{res}}$ but only learns $\mathsf{tct}_{\mathsf{res}}$. Simulation security of ThPKE is then sufficient to prove simulation security of the protocol. $\mathsf{IND\text{-}CPA}$ security of FHE is still needed to guarantee that the server does not learn anything about the parties' inputs. Note however that without circuit-privacy, the transcryptor could learn information about the parties' inputs when decrypting $\mathsf{fhct}_{\mathsf{res}}$ as the decryption error could depend on the underlying plaintexts.

C About Threshold FHE, IND-CPA-D Security, Circuit-Privacy, and More

To conclude our work, in this section, we introduce an indistinguishability-based security notion for Threshold FHE, which is implied by our simulation-based notion. Then we discuss its links to $\mathsf{IND\text{-}CPA\text{-}D}$ security and some mild forms of circuit-privacy.

C.1 Indistinguishability-Based Security for Threshold FHE

Below, we provide an indistinguishability-based security notion for threshold FHE, termed $\mathsf{Th\text{-}IND\text{-}CPA}$-security. Indistinguishability definitions for ThFHE have been proposed in prior works (e.g., in [JRS17,BS23a,KS23]). Our definition is similar in flavour though it is slightly more general. In particular, it is multi-hop and adaptive.

Definition C.1 (Threshold-IND-CPA Security). *We say that a threshold FHE scheme* $\mathsf{ThFHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{ServerDec}, \mathsf{PartDec}, \mathsf{FinDec})$ *is* Q_D-$\mathsf{Th\text{-}IND\text{-}CPA}$ *secure, if for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *making at most* Q_D *decryption queries, we have:*

$$\left|\Pr[\mathcal{A}(\mathsf{Exp}_1^{\mathsf{Th\text{-}IND\text{-}CPA}}(1^\lambda, 1^N)) = 1] - \Pr[\mathcal{A}(\mathsf{Exp}_0^{\mathsf{Th\text{-}IND\text{-}CPA}}(1^\lambda, 1^N)) = 1]\right| \leq \mathsf{negl}(\lambda),$$

where the experiment is described in Figure 6.

Lemma C.2. *Let* ThFHE *be a simulation-secure threshold FHE scheme. Then,* ThFHE *is* $\mathsf{Th\text{-}IND\text{-}CPA}$ *secure.*

Proof. For any $\mathsf{Th\text{-}IND\text{-}CPA}$ adversary, one can run it by replacing replies to its decryption queries by simulated answers using the simulator for ThFHE. Then, any adversary having non-negligible advantage contradicts simulation security of ThFHE. □

```
$\mathsf{Exp}_b^{\mathsf{Th\text{-}IND\text{-}CPA}}(1^\lambda, 1^N)$:
1: $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^N)$
2: $\mathsf{ctr} \leftarrow 0, \mathsf{L} \leftarrow \emptyset$
3: $(\mathcal{S}, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pk})$ with $\mathcal{S} \subset [N]$ and $|\mathcal{S}| < N$;
4: $b \leftarrow \mathcal{U}(\{0,1\})$
5: $b' \leftarrow \mathcal{A}_1^{\mathsf{OEnc}, \mathsf{OChall}_b, \mathsf{OEval}, \mathsf{ODec}}(\mathsf{pk}, (\mathsf{sk}_i)_{i \in \mathcal{S}}, \mathsf{st})$
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 $\mathsf{ct}$
```

```
$\mathsf{OChall}_b(\mu_0, \mu_1)$:
1: If $|\mu_0| \neq |\mu_1|$:
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 $\mathsf{ct}$
```

```
$\mathsf{OEval}(\mathsf{C}, (i_1, \ldots, i_\ell))$:
1: For $j \in [\ell]$:
2:     $(\mu_0, \mu_1, \mathsf{ct}_j) \leftarrow \mathsf{L}[i_j]$
3: $\mathsf{ct} \leftarrow \mathsf{Eval}(\mathsf{ek}, \mathsf{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$
4: $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
5: $\mu_0 \leftarrow \mathsf{C}(\mu_{0,1}, \ldots, \mu_{0,\ell})$
6: $\mu_1 \leftarrow \mathsf{C}(\mu_{1,1}, \ldots, \mu_{1,\ell})$
7: $\mathsf{L}[\mathsf{ctr}] \leftarrow (\mu_0, \mu_1, \mathsf{ct})$
8: Return $\mathsf{ct}$
```

```
$\mathsf{ODec}(j)$:
1: If $j > \mathsf{ctr}$:
2:     Return $\perp$
3: $(\mu_0, \mu_1, \mathsf{ct}) \leftarrow \mathsf{L}[j]$
4: If $\mu_0 \neq \mu_1$:
5:     Return $\perp$
6: $\mathsf{ct}_{\mathsf{dec}} \leftarrow \mathsf{ServerDec}(\mathsf{pk}, \mathsf{ct})$
7: $\mathsf{p}_k \leftarrow \mathsf{PartDec}(\mathsf{sk}_k, \mathsf{ct}_{\mathsf{dec}})$, for $k \in [N]$
8: Return $(\mathsf{ct}_{\mathsf{dec}}, (\mathsf{p}_k)_{k \in [N]})$
```

Fig. 6. Q_D-Th-IND-CPA security game for Threshold FHE.

C.2 About IND-CPA-D security

IND-CPA-D security was introduced in [LM21] as an extension of IND-CPA security in which the adversary can obtain decryption of honestly generated ciphertexts. While equivalent to IND-CPA security for perfectly correct schemes, IND-CPA-D security is not as easily satisfied by approximate FHE schemes (e.g., CKKS). Two recent works [CCP+24,CSB+24] also proved that statistical/perfect correctness is crucial even for non-approximate schemes (e.g., BGV, B/FV, DM/CGGI), as attacks can be mounted when decryption failure probability is non-negligible.

We do not recall the definition of IND-CPA-D security, but simply observe that it is identical to the definition of Th-IND-CPA-security for the specific case of $(1,1)$-threshold FHE, i.e. to the case where the secret key is not shared and kept as a whole. One can simply define FHE.Dec(sk, ct) as:

$$\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{ct}) = \mathsf{Combine}(\mathsf{PartDec}(\mathsf{sk}, \mathsf{ServerDec}(\mathsf{pk}, \mathsf{ct}))) ,$$

where $\mathsf{Combine}$ is actually a void algorithm since the secret key is not shared, so we can actually even write:

$$\mathsf{FHE.Dec}(\mathsf{sk},\mathsf{ct}) = \mathsf{PartDec}(\mathsf{sk},\mathsf{ServerDec}(\mathsf{pk},\mathsf{ct})) \ .$$

C.3 Relation with circuit-privacy

Our notion of $\mathsf{Th\text{-}IND\text{-}CPA}$ secure ThFHE (and therefore $\mathsf{IND\text{-}CPA\text{-}D}$ secure FHE as well via the observation above) implies a form of (computational) circuit privacy. Let us first define our notion of (weak-indistinguishability-based) circuit-privacy. For simplicity, we focus on non-threshold FHE (i.e. starting with an $\mathsf{IND\text{-}CPA\text{-}D}$ secure FHE scheme or equivalently a $(1,1)$-$\mathsf{Th\text{-}IND\text{-}CPA}$ secure ThFHE scheme) but the definition and construction easily generalizes to threshold FHE.

Definition C.3 (w-IND-CP Security). *Let λ denote a security parameter. We say that an FHE scheme* $\mathsf{FHE} = (\mathsf{KeyGen},\mathsf{Enc},\mathsf{Eval},\mathsf{Dec})$ *is* $\mathsf{w\text{-}IND\text{-}CP}$ *secure, if for all PPT adversaries $\mathcal{A}$ making at most Q_D decryption queries, we have:*

$$\left|\Pr[\mathcal{A}(\mathsf{Exp}_1^{\mathsf{w\text{-}IND\text{-}CP}}(1^\lambda)) = 1] - \Pr[\mathcal{A}(\mathsf{Exp}_0^{\mathsf{w\text{-}IND\text{-}CP}}(1^\lambda)) = 1]\right| \le \mathsf{negl}(\lambda)$$

where the experiment is described in Figure 7.

We talk about *weak* circuit-privacy as the adversary only gets access to a decryption oracle, while for (strong) circuit-privacy, it is common to provide the secret key (or the randomness used as input of KeyGen to generate it) to the adversary (e.g., as in Definition B.1).

We now explain how to achieve this security notion. For simplicity, we consider binary plaintexts. Denote $\mathsf{ct}_1,\ldots,\mathsf{ct}_\ell$ the ciphertexts on which one wants to evaluate homomorphically a circuit C of size s. We denote by $\mu_1,\ldots,\mu_\ell$ the underlying bits of $\mathsf{ct}_1,\ldots,\mathsf{ct}_\ell$. We proceed as follows:

1. Compute a binary representation $(\mathsf{C}_1|\cdots|\mathsf{C}_s) \in \{0,1\}^s$ of C;
2. Encrypt all bits $\mathsf{C}_1,\ldots,\mathsf{C}_s$ as $\mathsf{ct}_{\mathsf{C},i} \leftarrow \mathsf{Enc}(\mathsf{pk},\mathsf{C}_i)$ for $i \in [s]$;
3. Compute $\mathsf{ct} \leftarrow \mathsf{Eval}(U_{s,\ell},\mathsf{ct}_{f,1},\ldots,\mathsf{ct}_{\mathsf{C},s},\mathsf{ct}_1,\ldots,\mathsf{ct}_\ell)$, where $U_{s,\ell}$ denotes the universal circuit for circuits of size s with ℓ inputs, so that
$$U_{s,\ell}(\mathsf{C}_1,\ldots,\mathsf{C}_s,\mu_1,\ldots,\mu_\ell) = \mathsf{C}(\mu_1,\ldots,\mu_\ell) \ .$$
4. Return ct.

Correctness of the evaluation immediately follows from correctness of the FHE scheme and by definition of universal circuits. For security, since the circuits are now encrypted, $\mathsf{Th\text{-}IND\text{-}CPA}$ or $\mathsf{IND\text{-}CPA\text{-}D}$ security guarantees that, even given access to decryption queries, it is computationnally hard to distinguish the evaluation of any two circuits $\mathsf{C}_0,\mathsf{C}_1$ of the same size as long as $\mathsf{C}_0(\mu_1,\ldots,\mu_\ell) = \mathsf{C}_1(\mu_1,\ldots,\mu_\ell)$, since circuits of the same size are evaluated using the same universal circuit. Therefore, any $\mathsf{Th\text{-}IND\text{-}CPA}$ secure ThFHE (resp. $\mathsf{IND\text{-}CPA\text{-}D}$ secure FHE) scheme induces a Threshold FHE (resp. FHE) scheme satisfying $\mathsf{w\text{-}IND\text{-}CP}$.

```
$\mathsf{Exp}_b^{\mathsf{w\text{-}IND\text{-}CP}}(1^\lambda, N, t)$:
 1: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
 2: $\mathsf{ctr} \leftarrow 0, L \leftarrow []$
 3: $b \leftarrow \mathcal{U}(\{0,1\})$
 4: $b' \leftarrow \mathcal{A}^{\mathsf{OEnc},\mathsf{OEvalChall}_b,\mathsf{ODec}}(\mathsf{pk})$
 5: 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 $\mathsf{ct}$

$\mathsf{OEvalChall}_b(\mathsf{C}_0, \mathsf{C}_1, (i_1, \ldots, i_\ell))$:
 1: If $|\mathsf{C}_0| \neq |\mathsf{C}_1|$:
 2:     Return $\perp$
 3: For $j \in [\ell]$:
 4:     $\mu_{0,j} \leftarrow \mathsf{L}[i_j][0]$
 5:     $\mu_{1,j} \leftarrow \mathsf{L}[i_j][1]$
 6:     $\mathsf{ct}_j \leftarrow \mathsf{L}[i_j][2]$
 7: $\mathsf{ct} \leftarrow \mathsf{Eval}(\mathsf{pk}, \mathsf{C}_b, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$
 8: $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
 9: $\mu_0 \leftarrow \mathsf{C}_0(\mu_{0,1}, \ldots, \mu_{0,\ell})$
10: $\mu_1 \leftarrow \mathsf{C}_1(\mu_{1,1}, \ldots, \mu_{1,\ell})$
11: $\mathsf{L}[\mathsf{ctr}] \leftarrow (\mu_0, \mu_1, \mathsf{ct})$
12: Return $\mathsf{ct}$
```

```
$\mathsf{ODec}(j)$:
 1: If $j > \mathsf{ctr}$:
 2:     Return $\perp$
 3: $\mu_0 \leftarrow \mathsf{L}[j][0]$
 4: $\mu_1 \leftarrow \mathsf{L}[j][1]$
 5: $\mathsf{ct} \leftarrow \mathsf{L}[j][2]$
 6: If $\mu_0 \neq \mu_1$:
 7:     Return $\perp$
 8: $\mu \leftarrow \mathsf{Dec}(\mathsf{pk}, \mathsf{ct})$
 9: Return $\mu$
```

Fig. 7. w-IND-CP security game.

C.4 About One-Way-CPA security vs IND-CPA security

In this last section, we explain that OW-CPA security for (standard) FHE does not imply IND-CPA security in general, but does as soon as the scheme satisfies a mild form of circuit-privacy. Again, as mentioned earlier, we emphasize that this only holds for basic IND-CPA security. It seems very challenging to extend the technique to security notions involving decryption queries, such as IND-CPA-D security or threshold security. We do not remind the definition of OW-CPA nor IND-CPA security.

Lemma C.4. *There exists a* OW-CPA *FHE scheme which is not* IND-CPA *secure.*

Proof. Consider an IND-CPA secure FHE scheme with message space $\{0,1\}^\lambda$. Then, consider the scheme FHE' obtained by simply modifying the previous scheme by changing the encryption algorithm such that $\mathsf{Enc}_{\mathsf{pk}}(0^\lambda)$ returns 0.

The resulting scheme is clearly not IND-CPA, since it is easy to distinguish an encryption of 0^λ from any other ciphertext. Yet, it remains OW-CPA since with overwhelming probability, the challenge is not 0^λ and IND-CPA security of the original scheme guarantees OW-CPA security.

Moreover, the scheme is still fully homomorphic with the following modified Eval′ algorithm. For addition, if neither of the input ciphertexts is 0, run Eval, otherwise, addition ignores the zero ciphertext returns the other ciphertext. For multiplication, if one of the input ciphertexts is 0, return 0. □

Now, assume we have a OW-CPA FHE scheme with message space $\{0,1\}^\ell$ which satisfies circuit-privacy for the very specific family of functions $C_{i,\mu^{(0)},\mu^{(1)}}$ for $i \in [\ell]$ and $\mu^{(0)}, \mu^{(1)} \in \{0,1\}^\ell$ which on input a message $\mu \in \{0,1\}^\ell$, returns $\mu^{(\mu_i)}$ where μ_i denotes the i-th bit of μ. This is simply a MUX operation. Then we have the following.

Theorem C.5. *Assuming the above, this* OW-CPA *scheme is also* IND-CPA *secure.*

The proof relies on the splitting lemma, which we remind below.

Lemma C.6 (Splitting Lemma). *Let $A \subseteq X \times Y$ such that $\Pr[(x,y) \in A] \geq \varepsilon$. For any $\varepsilon' < \varepsilon$, defining B as $B = \{(x,y) \in X \times Y \mid \Pr_{y' \leftarrow U(Y)}[(x,y') \in A]\} \geq \varepsilon - \varepsilon'$, then we have:*

(i) $\Pr[B] \geq \varepsilon'$
(ii) $\forall (x,y) \in B, \Pr_{y'}[(x,y') \in A] \geq \varepsilon - \varepsilon'$
(iii) $\Pr[B \mid A] \geq \varepsilon'/\varepsilon$

Proof. Let $\mathcal{A}$ denote an adversary against the IND-CPA security of the scheme. Then, we construct an adversary $\mathcal{B}$ against its OW-CPA security as follows.

$\mathcal{B}$ receives from its challenger an encryption ct^* of a uniformly random $\mu^* \in \{0,1\}^\lambda$.

The main idea is to run $\mathcal{A}$ (several times) in order to recover each bit of μ^* by using μ^*_i as the random bit for the IND-CPA challenger. Let us focus on recovering the first bit of μ^*, denoted μ^*_1.

$\mathcal{B}$ runs $\mathcal{A}$. It first forwards it the public parameters (including the fixed public key, if we are in the public key setting).

When $\mathcal{A}$ makes a challenge query $(\mu^{(0)}, \mu^{(1)})$, $\mathcal{B}$ evaluates $C_{1,\mu^{(0)},\mu^{(1)}}$ homomorphically on its challenge ciphertext ct^* and forwards the resulting ciphertext to $\mathcal{A}$.

Due to the circuit-privacy of the scheme, $\mathcal{B}$ simulates perfectly the challenger. When $\mathcal{A}$ halts with some prediction b', $\mathcal{B}$ stores b' as a prediction for μ^*_1. If $\mathcal{A}$'s advantage is ε, the $\mathcal{B}$'s guess is valid with probability ε.

The final ingredient to correctly predict the actual value of μ^*_1 is to run $\mathcal{A}$ several times until the guess is correct with high-probability. One subtlety is that the key is fixed here, but the Splitting Lemma (Lemma C.6) guarantees that, even with a fixed key, with probability $1/2$ over the choice of the key, $\mathcal{A}$ is a distinguisher with advantage at least $\varepsilon/2$ which can be restarted many times. Hence, if $\varepsilon = 1/\mathsf{poly}(\lambda)$, running $\mathcal{A}$ polynomially many times allows to recover μ_1^* with overwhelming probability, and we can repeat the process for every other bit of μ^*.

At the end of the process, $\mathcal{B}$ returns its guess for μ^*. $\square$