Threshold FHE with Short Decryption Shares without a Semi-trusted Server

Hiroki Okada[1,2] and Tsuyoshi Takagi[2]

[1] KDDI Research, Inc., Japan
[2] The University of Tokyo, Japan

Abstract. Threshold fully homomorphic encryption (ThFHE) enables decryption by collecting *decryption shares* from any T-out-of-N parties. A major drawback of previous ThFHE schemes is that they require a super-polynomial modulus (or are subject to other limitations), resulting in long ciphertexts, keys, and decryption shares. Passelègue and Stehlé (Asiacrypt 2024) proposed a ThFHE scheme in which a *semi-trusted* server rounds the input ciphertexts to produce polynomially short ciphertexts and sends them to the parties, thereby making the rest of the decryption process efficient. Although the input ciphertexts are still super-polynomially large, the communication cost of sending them from the parties to the server can be reduced to polynomial size via the *transciphering* technique; as a result, an entirely low-communication ThFHE is achieved. However, if even a single party colludes with the server (contrary to the assumption), the secret key can be efficiently recovered. Such a risky scenario would be unsuitable for practical deployment.

In this paper, we tackle this issue. We propose two *serverless* ThFHE schemes with *polynomially short decryption shares*. The core idea is to let the parties directly round the decryption shares, rather than rely on the semi-trusted server to round the ciphertexts. We can also achieve low-communication ThFHE by reducing the communication required to send input ciphertexts to the parties to polynomial size via transciphering. Our first scheme, based on binary coefficient linear secret sharing ($\{0,1\}$-LSS), strictly improves upon Boneh et al. (CRYPTO 2018), achieving short decryption shares without any trade-offs. Our second scheme, based on Shamir secret sharing, adapts the technique of Okada and Takagi (Asiacrypt 2025) to eliminate the $O(N^{4.3})$ overhead in share size of our first scheme, further reducing communication costs.

Keywords: Lattice · FHE · Threshold Cryptography

1 Introduction

Fully homomorphic encryption (FHE) [Gen09] enables arbitrary computation on encrypted data without the decryption key. Threshold FHE (ThFHE) extends

This preprint is the full version of the paper that will appear in the proceedings of ACISP 2026 [OT26b]. See Appendix A for a summary of the changes.

FHE by enabling decryption through the collection of *partial decryption shares* from T-out-of-N parties. The U.S. National Institute of Standards and Technology (NIST) launched a project to standardize threshold cryptographic systems such as threshold signatures, threshold public-key encryption (PKE), and ThFHE in 2023 [BP25]. The primary application of ThFHE involves multiparty computation (MPC) protocols, and it has been shown to achieve round-optimal constructions [GGHR14; GLS15]. In addition, ThFHE enables the construction of a universal thresholdizer [BGG+18], which provides threshold functionality for cryptosystems such as signatures, (CCA-secure) PKE, pseudorandom functions (PRFs), and functional encryption.

ThFHE was first presented by Asharov et al. [AJL+12], and subsequently, Boneh et al. [BGG+18] extended it to allow arbitrary T-out-of-N decryption. An inherent drawback of their construction is that it requires a super-polynomial modulus, which leads to super-polynomially long keys, ciphertexts, and decryption shares. Although Boudgoust and Scholl [BS24] presented a ThFHE scheme with a polynomially small modulus, it achieves only *selective* security: the adversary may query ciphertexts only before receiving the challenge ciphertexts. Passelègue and Stehlé [PS25] presented a simulation-secure ThFHE scheme in which a "semi-trusted" server rounds off the input ciphertexts to produce polynomially short ciphertexts and sends them to parties, making the rest of the decryption process efficient. Although the modulus of the input ciphertexts is still super-polynomially large, the communication cost required to send them to the server, for example from the parties, can be reduced to polynomial size via the *transciphering* technique [NLV11; BCK+23], at the expense of the computational cost of the homomorphic evaluation of the decryption algorithm of symmetric key encryption, such as AES or "FHE-oriented" schemes [ARS+15; HKL+22]. As a result, [PS25] achieves entirely low-communication ThFHE.

However, the scheme of [PS25] imposes a strong requirement. The standard security of ThFHE requires that the adversary corrupts at most $T-1$ parties. In addition, the security proof of [PS25] requires the existence of a special "semi-trusted" party called a server, which 1) is not corrupted by the adversary and 2) does not receive partial decryption shares. Indeed, if the adversary corrupts even a single party and the server contrary to the requirements, the adversary can efficiently recover the FHE secret key. That is, if only one party colludes with the server, and sends partial decryption shares, then the party and the server can recover the secret key. Since this simple attack is not shown in [PS25] and elsewhere, we present it in Sect. 2.3. Such a "risky" situation would not be desirable in practical deployment settings. A follow-up work by Okada and Takagi [OT26a] also relies on the semi-trusted server.

Recently, Zyskind et al. [ZZLP25] proposed a ThFHE scheme with a polynomial modulus and a low-communication decryption procedure. Their approach differs significantly from previous work. They perform the decryption algorithm efficiently using MPC in the arithmetic black-box (ABB) model [DPSZ12], leveraging precomputation. The cost model of [ZZLP25] also departs from prior works: although the online phase is highly efficient, the offline phase

requires parties to precompute and store a large number of gates, amounting to, e.g., 79 MB of storage for 1000 decryption queries. Moreover, the online decryption protocol involves 5 rounds of communication and is therefore not round-optimal. Hence, the following open problem exists:

Question 1.1. Can we construct a (simulation-secure) ThFHE scheme that achieves polynomially short decryption shares *without a semi-trusted server* (and without large precomputed gate storage)?

Another major limitation of existing ThFHE schemes (except for the MPC-based one in [ZZLP25]) is that they rely on binary-coefficient linear secret sharing ($\{0,1\}$-LSS) [BGG+18] to realize arbitrary T-out-of-N threshold decryption. The use of $\{0,1\}$-LSS significantly increases the size of secret shares and decryption shares by a factor of $O(N^{4.3})$ [Val84; Gol20]. A natural choice for constructing threshold decryption is Shamir's secret sharing (Shamir-SS), whose share size is constant w.r.t. N. Indeed, a Shamir-SS-based ThFHE scheme is presented in [BGG+18, Constr. 5.11]. However, the scheme requires a super-exponentially ($=\omega((N!)^3)$) large modulus because the large Lagrange coefficients significantly increase the noise term. This drawback was mitigated by Okada and Takagi [OT26a], who proposed efficient Shamir-SS-based ThFHE schemes with short decryption shares. However, their scheme instead has other limitations. They proposed two types of schemes: 1) the first scheme is not tightly reduced from the LWE assumption, and 2) the second scheme requires a bound on the number of decryption queries, and enlarges the secret shares by a factor of $O(L_{\mathsf{Dec}})$, where L_{Dec} is the query bound. In addition, both schemes require a semi-trusted server, as in [PS25]. Hence, the following problem remains open:

Question 1.2. Can we construct a Shamir-SS-based (simulation-secure) ThFHE with polynomially short decryption shares that is tightly reduced from LWE and with an unbounded number of decryption queries?

1.1 Our contributions

We address both Questions 1.1 and 1.2 by proposing two ThFHE schemes with polynomially short decryption shares *without a semi-trusted server*: one via $\{0,1\}$-LSS and the other via Shamir-SS.

We summarize our contributions in Table 1 and compare them with previous works. In short, our first construction (in §5) strictly improves [BGG+18, Construction 5.6] by reducing the decryption shares to polynomial size without incurring any trade-off. Our second construction (in §6) extends the first construction to support Shamir-SS, further reducing the share size by a factor of $O(N^{4.3})$ at the cost of one additional communication round.

Serverless ThFHE with short decryption shares (§5). We present a ThFHE scheme with polynomially short decryption shares without a semi-trusted server based on $\{0,1\}$-LSS in Sect. 5, which addresses Question 1.1.

Table 1. Comparison of ThFHE schemes. Drawbacks are highlighted. The "Sharing" row indicates the secret sharing method, which is $\{0,1\}$-LSS, Shamir-SS, or MPC in the ABB model. The "Rounds" row indicates the communication rounds of the threshold decryption procedure. "Enc mod. Q" and "Dec mod. q" denote the modulus size of the input ciphertexts (Q) and decryption shares (q), respectively. For the "Dec share" and "sk share" rows, we write the number of elements included in decryption and secret share vectors, respectively. L_{Dec} denotes the decryption query bound, and we write $L_{\mathsf{Dec}} := \infty$ if it is unbounded. The "Mod/noise" row represents the modulus-to-noise ratio of the underlying LWE assumption. The "Sim. Sec." row indicates schemes that achieve simulation security. "–" denotes NO. The "Tight" row indicates schemes that achieve tight security. The "W/o server" row indicates schemes that avoid relying on a *semi-trusted* server, which is assumed to be semi-honest and uncorrupted.

	[BGG+18] (§5.2*)	[BS24]	[PS25]	[OT26a] (§6)	[ZZLP25]	Ours §5	Ours §6
Sharing	$\{0,1\}$-LSS	$\{0,1\}$-LSS	$\{0,1\}$-LSS	Shamir	ABB	$\{0,1\}$-LSS	Shamir
Rounds	1	1	2	3	5	1	2
Enc mod. Q	$\lambda^{\omega(1)\dagger}$	$\lambda^{O(1)}$	$\lambda^{\omega(1)\dagger\ddagger}$	$\lambda^{\omega(1)\dagger\ddagger}$	$\lambda^{O(1)}$	$\lambda^{\omega(1)\dagger}$	$\lambda^{\omega(1)\dagger}$
Dec mod. q	$\lambda^{\omega(1)}$	$\lambda^{O(1)}$	$\lambda^{O(1)}$	$\lambda^{O(1)}$	$\lambda^{O(1)}$	$\lambda^{O(1)}$	$\lambda^{O(1)}$
Dec share	$O(N^{4.3})$	$O(N^{4.3})$	$O(N^{4.3})$	1	NA¶	$O(N^{4.3})$	1**
sk share				$O(L_{\mathsf{Dec}})$			
Mod/noise	$\lambda^{\omega(1)}$	$\lambda^{O(1)}$	$\lambda^{\omega(1)}$	$\lambda^{\omega(1)}$	$\lambda^{O(1)}$	$\lambda^{\omega(1)}$	$\lambda^{\omega(1)}$
Query L_{Dec}	∞	Bounded	∞	Bounded	Bounded	∞	∞
Sim. Sec.	Yes	–§	Yes	Yes	Yes	Yes	Yes
Tight	Yes	–	–	Yes	Yes	Yes	Yes
W/o server	Yes	Yes	–	–	Yes	Yes	Yes

* Although [BGG+18, §5.3] provides also Shamir-SS-based scheme, we omit a description here, as it requires a super-exponentially $\omega((N!)^3)$ large modulus.

† The transciphering technique [NLV11; BCK+23] can be used to reduce the communication size of super-polynomially long FHE ciphertexts to polynomial sizes.

‡ Although the input ciphertexts sent to the server are super-polynomially large, the ciphertexts sent to the parties are polynomially small (without transciphering).

¶ [ZZLP25] has no concept of secret/decryption share. Instead, we note here that their online communication size is 656 bits per 1 decryption query, and they need precomputed "gate storage" of 79 MB per 1000 decryption queries.

§ Only the *selective* security is achieved because of the cryptanalysis of [PS25].

** The entry counts only algebraic secret-share elements; the Shamir-based construction additionally stores $O(N)$ PRF seeds per party for zero-share masking.

In previous work [PS25], the server performs a randomized rounding operation, denoted by $\mathsf{ServerDec}$, on the input ciphertexts with its own randomness to produce polynomially short ciphertexts. Hence, to keep the server's randomness hidden from the adversary, the server must be "semi-trusted," i.e., assumed to remain uncorrupted by the adversary. Our construction is simpler: the parties directly round their partial decryption using their own randomness, without

relying on a server. The security proof then relies on the randomness of honest parties rather than that of a semi-trusted server.

Furthermore, our scheme improves upon [PS25] in terms of round complexity and underlying computational assumption. The threshold decryption in [PS25] requires 2 rounds of communication. In the 1st round, the server performs a rounding procedure and broadcasts the output to the parties. In the 2nd round, the parties generate partial decryption shares and broadcast them to all parties, but not to the server. In contrast, our construction requires only 1 round of communication, as in other previous works [AJL+12; BGG+18; BS24]; all of the parties generate partial decryption shares on the input ciphertext and broadcast them in the 1st round. With respect to the computational assumption, [PS25] is based on "yet another" variant of LWE (yaLWE). Although reductions from LWE to yaLWE are known [MS25; BJTW25], they are not tight. In contrast, our schemes are directly based on the standard LWE[3], and the security reduction is tight. As a result, our construction strictly improves [BGG+18, Constr. 5.6], achieving polynomially short decryption shares without any trade-offs.

Extension to Shamir-SS-based ThFHE (§6). Next, we extend our $\{0,1\}$-LSS-based construction to an efficient Shamir-SS-based ThFHE scheme by adapting the *synchronous* technique of Okada and Takagi [OT26a] to our serverless setting, which addresses Question 1.2.

We compare our Shamir-SS-based construction with the ThFHE schemes by Okada and Takagi [OT26a]. As mentioned, [OT26a] proposed two schemes with limitations: 1) the first scheme is not tightly reduced from LWE, and 2) the second scheme requires a bound on the number of decryption queries and increases the size of secret shares by a factor of the query bound $O(L_{\mathsf{Dec}})$. Our Shamir-SS-based scheme removes these limitations, and does not require a semi-trusted server. Furthermore, compared to schemes in [OT26a], our construction reduces the number of communication rounds by one. The threshold decryption in [OT26a] is a 3-round protocol; during the 1st round, each party declares its participation (active/inactive) in the decryption session[4], i.e., the protocol requires one round for "synchronization". The remaining protocol is the same as that of [PS25]; in the 2nd round, the server rounds off the input ciphertext and broadcasts the output to the parties, and in the 3rd round, each party generates the decryption share and broadcasts it to all parties except the server. The threshold decryption procedure of our Shamir-SS-based scheme is a 2-round protocol: In the 1st round, each party declares its participation. In the 2nd round, each party generates the decryption share and broadcasts it. As a result, our second scheme extends the first one to Shamir-SS-based construction, which

[3] More precisely, our ThFHE does not rely on the LWE assumption; it is only required to establish the semantic security of the underlying FHE.

[4] Although [OT26a] assumes the active party set is somehow known to all parties, in practice, this requires one additional round of communication to declare participation.

reduces the share size by a factor of $O(N^{4.3})$, at the cost of an additional communication round.

Note. In [PS25; OT26a], although the modulus of the input ciphertexts is still super-polynomially large, all communication to and from the parties is polynomially small. From the perspective of the parties, low-communication is achieved *even without using transciphering*. In contrast, in our serverless construction, the super-polynomially long input ciphertexts are directly sent to the parties; thus, we require transciphering to achieve low-communication. However, when applying ThFHE to construct MPC, the input ciphertexts are typically sent by the parties (see, e.g., [AJL+12]). Consequently, in such a setting, [PS25; OT26a] also requires transciphering to achieve low-communication MPC. Also note that in the server-aided setting [PS25; OT26a], homomorphic computation on super-polynomially large ciphertexts is performed by the server, whereas in the serverless setting [AJL+12; BGG+18] and in our construction, it is performed by the parties.

1.2 Future Work

Our ultimate goal is to achieve polynomial-size moduli for both input ciphertexts and decryption shares. While we achieve short decryption shares, the modulus for input ciphertexts in our scheme is still super-polynomially large. Although this drawback is mitigated by transciphering, it incurs the cost of homomorphically decrypting symmetric ciphertexts. Note that although [BS24; ZZLP25] achieve polynomial moduli, they have other drawbacks, as already described in Sect. 1.

Closely related to the above, while the security of [BS24; ZZLP25] can be reduced from the hardness of LWE with a polynomial modulus-to-noise ratio, other works, including ours, require a super-polynomial modulus-to-noise ratio. This requires assuming the hardness of LWE with a super-polynomial modulus-to-noise ratio and a larger LWE dimension n for concrete parameter selection. Nevertheless, many cryptographic applications based on LWE with a super-polynomial modulus-to-noise ratio have been proposed. In particular, it is worth noting that the BGV FHE scheme [BGV12] is based on LWE instances whose modulus-to-noise ratio is exponential in the number of levels L.

We leave it for future work to construct a simulation-secure ThFHE with entirely polynomial-size moduli without large precomputation storage.

1.3 Organization

The remainder of this paper is organized as follows: In Sect. 2, we provide a technical overview of our schemes. In Sect. 3, we describe necessary definitions and lemmas. In Sect. 4, we define FHE and ThFHE. Then, in Sect. 5, we address Question 1.1: we present a ThFHE scheme with polynomially short decryption shares *without a semi-trusted server* via $\{0,1\}$-LSS. Next, in Sect. 6, we address Question 1.2: we extend our first scheme into an efficient ThFHE scheme with Shamir-SS.

2 Technical Overview

In this section, we provide a technical overview of our schemes and previous works. We begin by describing the common settings in Sect. 2.1. Next, in Sect. 2.2, we recall the schemes from [AJL+12; BGG+18], whose constructions require a super-polynomially large modulus. We then discuss the ThFHE scheme of [PS25] in Sect. 2.3, which relies on a semi-trusted server to achieve a polynomially small modulus for the decryption shares. Finally, we explain the core techniques of our schemes in Sect. 2.4 and Sect. 2.5.

2.1 Settings

Underlying FHE. For simplicity, we consider the underlying FHE to be based on LWE. We define the public key as $\mathsf{pk} := (\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e})$, with $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_Q^{m \times n}$, secret key $\mathbf{s} \in \mathbb{Z}_Q^n$, short secret error $\mathbf{e} \in \mathbb{Z}_Q^m$, and modulus Q. In addition, for simplicity, we use full threshold (N-out-of-N) decryption with simple additive secret sharing here, which is a special case of $\{0,1\}$-LSS: each party $P_1, \ldots, P_N$ owns a secret share $\mathbf{s}_1, \ldots, \mathbf{s}_N$, with the property that their sum equals the overall secret $\mathbf{s}$, i.e., $\sum_{i=1}^{N} \mathbf{s}_i = \mathbf{s}$.

We define the input ciphertexts, which may be the outputs of the homomorphic evaluation algorithm, as follows:

$$\mathsf{ct} := (\mathbf{a} \xleftarrow{\$} \mathbb{Z}_Q^n, b := \mathbf{a}^\intercal \mathbf{s} + e_{\mathsf{ct}} + \lfloor \tfrac{Q}{2} \rfloor \mu \bmod Q), \tag{1}$$

where e_{ct} is a (small) *ciphertext noise*, and $\mu \in \mathcal{M} := \{0,1\}$ is a plaintext.

Ciphertext noise contains secret information. Notably, the ciphertext noise e_{ct} depends on secret information. For example, in the (standard) LWE-based PKE from [LP11], the ciphertext noise e_{ct} is

$$e_{\mathsf{ct}} := \mathbf{r}^\intercal \mathbf{e} - \mathbf{f}^\intercal \mathbf{s} + f,$$

where $\mathbf{r}, \mathbf{f}$ and f represent the small randomness of the encryption algorithm. e_{ct} strongly depends on the secret key $\mathbf{s}$ and secret error $\mathbf{e}$, which is also the case with FHE. For example, in the FHE scheme called CKKS [CHK+18], the ciphertext noise of the product of two ciphertexts, ct_0 and ct_1, encrypting messages μ_0 and μ_1 with the respective noise terms e_0 and e_1, is given by the following equation:

$$e_{\mathsf{ct}} = \mu_0 e_{\mathsf{ct}_1} + \mu_1 e_{\mathsf{ct}_0} + e',$$

where e' is a small noise. The fact that e_{ct} depends on secret information such as secret key $\mathbf{s}$ and secret error $\mathbf{e}$, and input plaintexts μ_0 and μ_1 (in addition to the result plaintext $\mu_0 \cdot \mu_1$) is crucially problematic in threshold decryption, as we explain in the next subsection.

2.2 ThFHE with a super-polynomial modulus [AJL+12; BGG+18]

We recall the ThFHE schemes of [AJL+12; BGG+18]. Given ct, as defined in (1), each party generates a decryption share as defined below:

$$\mathsf{pd}_i \leftarrow \mathsf{PartDec}(\mathsf{ct}, \mathbf{s}_i) := \mathbf{a}^\intercal \mathbf{s}_i + E_i \bmod Q,$$

where E_i represents a (large) noise. The parties then broadcast pd_i to all participants. Using the set $\{\mathsf{pd}_i\}_{i\in[N]}$, each party calculates

$$\begin{aligned}\overline{\mu} := \mathsf{FinDec}(\mathsf{ct}, \{\mathsf{pd}_i\}_{i\in[N]}) &:= \lfloor (b - \textstyle\sum_{i\in[N]} \mathsf{pd}_i)/\lfloor \frac{Q}{2} \rfloor \rceil \\ &= \mu + \lfloor (e_{\mathsf{ct}} - \textstyle\sum_{i\in[N]} E_i)/\lfloor \frac{Q}{2} \rfloor \rceil.\end{aligned}$$

Then, μ is recovered if the parameters are chosen such that the noise term $e_{\mathsf{ct}} - \sum_{i=1}^{N} E_i$ satisfies $|\overline{e}| < \lfloor \frac{Q}{4} \rfloor$ with overwhelming probability. Importantly, the noise term can also be recovered by the parties via the following calculation:

$$e_{\mathsf{leak}} := b - \textstyle\sum_{i\in[N]} \mathsf{pd}_i - \lfloor \frac{Q}{2} \rfloor \mu = e_{\mathsf{ct}} - \textstyle\sum_{i\in[N]} E_i.$$

As noted in Sect. 2.1, the ciphertext noise e_{ct} contains secret information. This leakage of e_{ct} basically poses no issue[5] in single-key FHE (or PKE), because the decryption is allowed only to the party possessing the secret key $\mathbf{s}$ (and $\mathbf{e}$), who is authorized to recover the plaintext. However, this becomes problematic in ThFHE, because decryption is performed by parties who hold only a secret share $\mathbf{s}_i$, and the plaintext of ct (as the result of threshold decryption).

Noise flooding. Therefore, for security, undesirable information in e_{ct} must be hidden. That is, e_{ct} is not simulatable without the knowledge of $\mathbf{s}$ and/or $\mathbf{e}$. The "noise flooding" technique was presented by Asharov et al. [AJL+12, Lemma 2.1] to smudge out e_{ct} by adding super-polynomially large noise. We sample

$$E_i \xleftarrow{\$} [-B_{\mathsf{flood}}, B_{\mathsf{flood}}], \text{ where } B_{\mathsf{flood}} = \lambda^{\omega(1)} B_{\mathsf{err}} \text{ and } |e_{\mathsf{ct}}| < B_{\mathsf{err}}. \tag{2}$$

Then, E_i information-theoretically hides e_{ct};

$$e_{\mathsf{leak}} = e_{\mathsf{ct}} - \textstyle\sum_{i\in[N]} E_i \approx_{\mathsf{s}} - \sum_{i\in[N]} E_i.$$

A crucial drawback of noise flooding is that the modulus Q must be super-polynomially large ($Q = \lambda^{\omega(1)}$) to satisfy correctness. Hence, keys, shares, and ciphertexts must be super-polynomially long.

[5] As discussed in the IND-CPA^D line of work, high decryption failure probability or approximate decryption of [CHK+18] also pose some issues.

2.3 Low-communication ThFHE with the semi-trusted server [PS25]

Passelègue and Stehlé [PS25] modified the "noise flooding" technique into the "flood-and-round" algorithm, denoted as $\mathsf{ServerDec}$. On an input super-polynomially long ct, $\mathsf{ServerDec}$ first performs "noise flooding" to smudge out ciphertext noise e_{ct} and then performs "rounding" to output polynomially short ciphertext $\mathsf{ct}_{\mathsf{dec}}$:

$$\mathsf{ServerDec}(\mathsf{ct} \in \mathbb{Z}_Q^{n+1}) \to \mathsf{ct}_{\mathsf{dec}} \in \mathbb{Z}_q^{n+1}.$$

$\mathsf{ServerDec}$ must be performed by a "semi-trusted" party called Server, which is assumed not to be corrupted by the adversary.

"Flood-and-round". We technically explain the flood-and-round technique of [PS25]. Let ct in (1) be the input of $\mathsf{ServerDec}$. It first calculates $\mathsf{ct}_{\mathsf{flood}} := \mathsf{ct} + \mathsf{ct}_{\mathsf{fresh}} + (0, E)$, where E is a super-polynomially large Gaussian noise and $\mathsf{ct}_{\mathsf{fresh}}$ is a fresh encryption of 0 generated by pk and the randomness of the server. By the smudging lemma for Gaussians [PS25, Lemma 2.1], we have

$$\mathsf{ct}_{\mathsf{flood}} := (\mathbf{a}, \mathbf{a}^\intercal\mathbf{s} + e_{\mathsf{ct}} + E + \lfloor \tfrac{Q}{2} \rfloor \cdot \mu) \approx_{\mathsf{s}} (\mathbf{a}, \mathbf{a}^\intercal\mathbf{s} + E + \lfloor \tfrac{Q}{2} \rfloor \cdot \mu).$$

Then, the super-polynomial modulus Q, where $Q = p \cdot q$ for $p = \lambda^{\omega(1)}$ and $q = \lambda^{O(1)}$, is switched to a polynomially small modulus q by rounding;

$$\mathsf{ct}_{\mathsf{dec}} := \left(\left\lfloor \tfrac{1}{p} \cdot \mathbf{a} \right\rceil_{\sigma_0}, \left\lfloor \tfrac{1}{p} \cdot (\mathbf{a}^\intercal\mathbf{s} + E + \lfloor \tfrac{Q}{2} \rfloor \cdot \mu) \right\rceil_{\sigma_1} \right) := (\overline{\mathbf{a}}, \overline{b}),$$

where $\lfloor \cdot \rceil_\sigma$ represents a randomized Gaussian rounding (Def. 3.13) and σ_0 and σ_1 are parameters. We can rewrite the above equation as follows:

$$\begin{aligned} \mathsf{ct}_{\mathsf{dec}} &= (\overline{\mathbf{a}}, \overline{b} = \overline{\mathbf{a}}^\intercal\mathbf{s} + e_{\mathsf{dec}} + \lfloor \tfrac{q}{2} \rfloor \cdot \mu \bmod q), \\ \text{where } e_{\mathsf{dec}} &= \left\lfloor -\mathbf{r}^\intercal\mathbf{s} + \tfrac{1}{p}E \right\rceil_{\sigma_1} \approx_{\mathsf{s}} D_{\mathbb{Z}, \sqrt{\sigma_0^2\|\mathbf{s}\|^2 + \sigma_1^2}}, \end{aligned} \tag{3}$$

where $\|\cdot\|$ is the ℓ_2-norm and $\mathbf{r} := \overline{\mathbf{a}} - \frac{1}{p}\mathbf{a} \in \frac{1}{p}\mathbb{Z}^n$ is a randomized rounding error (cf., $\overline{\mathbf{a}} = \lfloor \frac{1}{p}\mathbf{a} \rceil_{\sigma_0} = \frac{1}{p}\mathbf{a} + \mathbf{r}$).

The threshold decryption procedure is performed only on $\mathsf{ct}_{\mathsf{dec}}$, which is the output of $\mathsf{ServerDec}$. The decryption share of [PS25] is

$$\mathsf{pd}_i := \overline{\mathbf{a}}^\intercal\mathbf{s}_i + e_i, \tag{4}$$

and all the parties can recover the ciphertext noise as follows:

$$e_{\mathsf{leak}} := \overline{b} - \textstyle\sum_{i \in [N]} \mathsf{pd}_i - \lfloor \frac{q}{2} \rfloor \cdot \overline{\mu} = e_{\mathsf{dec}} - \sum_{i \in [N]} e_i. \tag{5}$$

However, as (3) shows, e_{dec} depends on $\|\mathbf{s}\|$. For this reason, [PS25] employs the *known-norm* variant of LWE. While the known-norm LWE has reductions from standard LWE [MS25; BJTW25], they are not tight. This limitation is mitigated by Okada and Takagi [OT26a], but their scheme instead requires a bound on the number of decryption queries, and increases the size of secret shares by a factor of $O(L_{\mathsf{Dec}})$, where L_{Dec} is the query bound.

Why is a "semi-trusted" server needed? We explain why [PS25; OT26a] require a semi-trusted server. By contradiction, let us assume that the server is not semi-trusted; we assume that the adversary corrupts the server and at least one party. Since the randomized rounding error $\mathbf{r}$ is known to the adversary, (3) does not hold; i.e., the distribution of e_{dec} is now conditioned on the given value of $\mathbf{r}$. Hence, the server must be semi-trusted for the security proof.

Furthermore, there is an efficient key-recovery attack by this adversary. The adversary also obtains $\{\mathsf{pd}_i\}_{i\in[N]}$ and thus recovers the noise term e_{leak} as in (5) since the adversary corrupts at least one party. Indeed, corrupting only one party suffices to obtain e_{leak} since all parties have $\{\mathsf{pd}_i\}_{i\in[N]}$. We rewrite (5) as follows:

$$pe_{\mathsf{leak}} = -p\mathbf{r}^\intercal\mathbf{s} + E + p(r - \textstyle\sum_{i\in[N]} e_i) \in \mathbb{Z},$$

where $r \in \frac{1}{p}\mathbb{Z}$ is a (randomized) rounding error of e_{dec}; we define $e_{\mathsf{dec}} = \left\lfloor -\mathbf{r}^\intercal\mathbf{s} + \frac{1}{p}E \right\rceil_{\sigma_1} := -\mathbf{r}^\intercal\mathbf{s} + \frac{1}{p}E + r$. When correctness holds, we have $e_{\mathsf{leak}} < q$ and thus $e_{\mathsf{leak}} = e_{\mathsf{leak}} \bmod q$; there is no modular reduction by q. We also have $p \cdot e_{\mathsf{leak}} = p \cdot e_{\mathsf{leak}} \bmod Q$ since $Q = p \cdot q$. Hence, the tuple

$$(-p\mathbf{r}, pe_{\mathsf{leak}} = -p\mathbf{r}^\intercal\mathbf{s} + E + p(r - \textstyle\sum_{i\in[N]} e_i)) \in \mathbb{Z}^n \times \mathbb{Z}$$

constitutes a sample of the *integer LWE* (ILWE) problem [BDE+18], which is known as an easy problem; the adversary can efficiently recover $\mathbf{s}$.

2.4 Ours: ThFHE with short decryption shares without the semi-trusted server (§5)

We now provide a technical overview of our "serverless" ThFHE with short decryption shares, which removes the need for the semi-trusted server. In our ThFHE, parties directly "round off" the decryption shares to make them polynomially short rather than relying on a server to round off the ciphertexts.

Our key approach: round off the shares. Our key approach is to directly perform "rounding" in $\mathsf{PartDec}$, outputting polynomially short decryption shares. See Figs. 1 and 2 for an illustration of the system overviews of both [PS25] and our scheme, respectively. In the structure of [PS25; OT26a], long ciphertexts are rounded off into short ciphertexts during $\mathsf{ServerDec}$ with the private randomness of the semi-trusted server. In contrast, in our scheme, "rounding" is performed during $\mathsf{PartDec}$ with the randomness of the parties. We only require the randomness of honest parties, instead of the randomness of the semi-trusted server.

We now provide more technical details. Let $\mathsf{ct} := (\mathbf{a}, b)$ as defined in (1) be the input ciphertext. We then construct partial decryption as follows:

$$\mathsf{pd}_i \leftarrow \mathsf{PartDec}(\mathsf{ct}, \mathbf{s}_i) := \lfloor \tfrac{1}{p}(\mathbf{a}^\intercal\mathbf{s}_i + E_i \bmod Q) \rceil \bmod q, \tag{6}$$

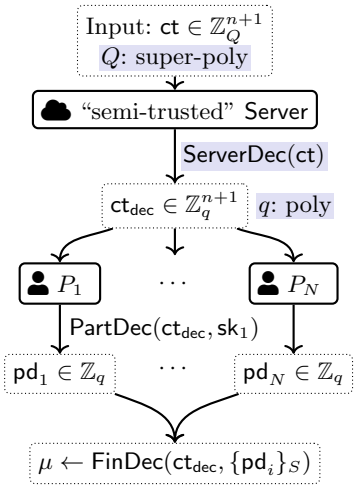


Fig. 1. Structure of [PS25; OT26a]: Super-polynomially long ciphertexts are rounded off into polynomially short ciphertexts via ServerDec performed by the semi-trusted server.

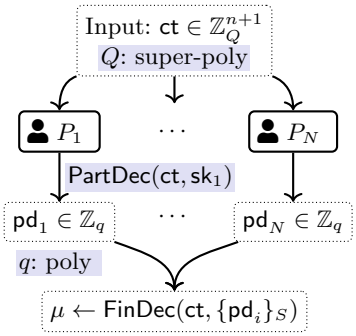


Fig. 2. Our structure: We directly perform "rounding" during PartDec, outputting polynomially short decryption shares.

where $E_i \xleftarrow{\$} [-B_{\mathsf{flood}}, B_{\mathsf{flood}}]$ as defined in (2). We also define

$$\begin{aligned}\mathsf{pd}_i &:= \tfrac{1}{p}\mathbf{a}^\intercal \mathbf{s}_i + \tfrac{1}{p}E_i - r_i, \text{ where } r_i := \{\tfrac{1}{p}(\mathbf{a}^\intercal \mathbf{s}_i + E_i \bmod Q)\} \\ &:= \tfrac{1}{p}\mathbf{a}^\intercal \mathbf{s}_i + e_i, \text{ where } e_i := \tfrac{1}{p}E_i - r_i.\end{aligned}$$

We also modify FinDec; parties locally round off the b part of ct during the FinDec procedure as follows;

$$\overline{\mu} \leftarrow \mathsf{FinDec}(\mathsf{ct}, \{\mathsf{pd}_i\}_{i\in[N]}) := \lfloor (\lfloor \tfrac{b}{p} \rfloor - \textstyle\sum_{i\in[N]} \mathsf{pd}_i \bmod q)/\lfloor \tfrac{q}{2} \rfloor \rceil.$$

By routine calculation and the fact that $\sum_{i\in[N]} \mathbf{a}^\intercal \mathbf{s}_i = \mathbf{a}^\intercal \mathbf{s}$, we obtain

$$\lfloor \tfrac{b}{p} \rfloor - \textstyle\sum_{i\in[N]} \mathsf{pd}_i \bmod q = \lfloor \tfrac{q}{2} \rfloor \cdot \mu + e' - \sum_{i\in[N]} e_i \bmod q,$$

where $e' := \frac{1}{p}e_{\mathsf{ct}} + (\{\frac{q}{2}\} - \frac{1}{p}\{\frac{Q}{2}\}) \cdot \mu - \{\frac{b}{p}\}$. Hence, the correctness holds by taking parameters such that the error term is small; i.e., $|e' - \sum_{i\in[N]} e_i| < \frac{q}{4}$.

Our "flood-and-round" method for simulating partial decryption. We use the "flood-and-round" technique of [PS25] for the security proof. While [PS25] uses the flood-and-round technique in the ServerDec algorithm to produce short ciphertexts, we use the technique to simulate a partial decryption share in the security proof.

We now (conservatively) assume that the adversary corrupts $N-1$ parties except for one party $h \in [N]$. The goal of the security proof is to simulate

a partial decryption pd_h of the honest party $h \in [N]$ using the view of corrupted $N-1$ parties, i.e., $\{\mathbf{s}_i\}_{i\in[N]\setminus h}$. Let $\mathsf{ct} := (\mathbf{a}, b := \mathbf{a}^\intercal\mathbf{s} + e_{\mathsf{ct}} + \lfloor \frac{Q}{2} \rfloor \cdot \mu)$ as in (1), and we simulate pd_h as follows;

$$\begin{aligned}\mathsf{pd}_h^{\mathsf{Sim}} &= \lfloor \tfrac{1}{p}(b - (\textstyle\sum_{i\in[N]\setminus\{h\}} \mathbf{a}^\intercal\mathbf{s}_i) + E_h - \lfloor \tfrac{Q}{2} \rfloor \cdot \mu \bmod Q) \rfloor \bmod q \\ &= \lfloor \tfrac{1}{p}(\mathbf{a}^\intercal\mathbf{s}_h + e_{\mathsf{ct}} + E_h \bmod Q) \rfloor \bmod q, \end{aligned} \tag{7}$$

where $E_h \xleftarrow{\$} [-B_{\mathsf{flood}}, B_{\mathsf{flood}}]$ as defined in (2), and we use the fact that $\mathbf{a}^\intercal\mathbf{s} - \sum_{i\in[N]\setminus\{h\}} \mathbf{a}^\intercal\mathbf{s}_i = \mathbf{a}^\intercal\mathbf{s}_h$. Furthermore, by the smudging lemma (Lem. 3.1) and the hypothesis of $B_{\mathsf{flood}} = \lambda^{\omega(1)} B_{\mathsf{err}}$ and $|e_{\mathsf{ct}}| < B_{\mathsf{err}}$ (as in (2)), we have

$$e_{\mathsf{ct}} + E_h \approx_{\mathsf{s}} E_h. \tag{8}$$

Thus, by (7) and (8), we obtain

$$\mathsf{pd}_h^{\mathsf{Sim}} \approx_{\mathsf{s}} \lfloor \tfrac{1}{p}(\mathbf{a}^\intercal\mathbf{s}_h + E_h \bmod Q) \rfloor,$$

and the right-hand side is identically distributed as real pd_h defined in (6). As described above, our approach employs "flooding" (smudging lemma) in the security proof, and furthermore achieves polynomial-size decryption shares by applying "rounding" to the decryption shares.

2.5 Our second scheme: Extension to Shamir-SS-based ThFHE (§6)

We first discuss the challenges in dealing with the Lagrange coefficients of Shamir-SS in threshold (homomorphic) encryption. Then, we briefly recall the ThFHE scheme of [OT26a], and explain our second scheme, which is based on Shamir-SS.

Difficulty of applying Shamir-SS to ThFHE. [BGG+18, Constr. 5.11] is a ThFHE scheme constructed with Shamir-SS (Constr. 3.7). The partial decryption share of the scheme is in the form

$$\mathsf{pd}_i \leftarrow \mathsf{PartDec}(\mathsf{ct}, \mathbf{s}_i) := \mathbf{a}^\intercal\mathbf{s}_i + (N!)^2 E_i.$$

The super-exponentially large factor $(N!)^2$ is required here for "clearing out" the large Lagrange coefficient [ABV+12]. For the sake of explanation, we now consider general T-out-of-N threshold decryption. *Active parties* $S_{\mathsf{act}} \subseteq [N]$ (s.t. $|S_{\mathsf{act}}| \geq T$) generate and broadcast pd_i; as a result, the parties obtain $\{\mathsf{pd}_i\}_{i\in S_{\mathsf{act}}}$. Then, each party freely chooses a valid subset $S \subseteq S_{\mathsf{act}}$ s.t. $|S| \geq T$ and locally executes FinDec as follows:

$$\begin{aligned}\mathsf{FinDec}(\mathsf{ct}, \{\mathsf{pd}_i\}_{i\in S}) &:= \lfloor (b - \textstyle\sum_{i\in S} L_i^S \cdot \mathsf{pd}_i \bmod Q) / \lfloor \tfrac{Q}{2} \rfloor \rceil \\ &= \mu + \lfloor (e_{\mathsf{ct}} - \textstyle\sum_{i\in S} L_i^S \cdot (N!)^2 E_i) / \lfloor \tfrac{Q}{2} \rfloor \rceil. \end{aligned}$$

However, this construction requires at least $Q = \Omega((N!)^3)$ to ensure correctness despite the large noise $\sum_{i \in S} L_i^S \cdot (N!)^2 E_i$, using the upper bound $L_i^S (N!)^2 < (N!)^3$ of [ABV+12], which makes the scheme impractical.

The security of [BGG+18, Constr. 5.11] is proved by simulating decryption shares of honest parties with corrupted secret shares, relying on the noise flooding. Thus, super-polynomially large E is used, i.e., $|E| = |e_{\mathsf{ct}}| \lambda^{\omega(1)}$. We now conservatively assume that the adversary corrupts the set of parties $S_{\mathsf{corr}} \subseteq [N]$ s.t. $|S_{\mathsf{corr}}| = T - 1$ (= maximal invalid set); let $S_{\mathsf{hon}} := [N] \setminus S_{\mathsf{corr}}$ be the set of honest parties, and let $S_{\mathsf{act}} := [N]$ for simplicity. For any $h \in S_{\mathsf{hon}}$, the challenger defines the set of indices $S_h := S_{\mathsf{corr}} \cup \{0, h\}$, calculates the Lagrange coefficient $L_i^{S_h}$, and simulates pd_h as follows:

$$\begin{aligned} \mathsf{pd}'_h &:= L_0^{S_h}(b - \lfloor \tfrac{Q}{2} \rfloor \mu) + \textstyle\sum_{i \in S_{\mathsf{corr}}} L_i^{S_h} \mathbf{a}^\intercal \mathbf{s}_i + (N!)^2 E \qquad (9) \\ &= \mathbf{a}^\intercal \mathbf{s}_h + L_0^{S_h} \cdot e_{\mathsf{ct}} + (N!)^2 E. \ (\because \text{by the correctness of Shamir-SS}). \end{aligned}$$

Then, we obtain $\mathsf{pd}'_h \approx_{\mathsf{s}} \mathbf{a}^\intercal \mathbf{s}_h + (N!)^2 E$, the right-hand side of which is identically distributed as the real pd_h, by the upper bound $(N!)^2 |L_0^{S_h}| < (N!)^3$ of [ABV+12] and the smudging lemma with the condition of $|E| = |e_{\mathsf{ct}}| \lambda^{\omega(1)}$.

Synchronous decryption with masking [OT26a]. The above difficulty is mitigated by Okada and Takagi [OT26a]. Their key approach is to "embed" the Lagrange coefficient in pd_i as follows:

$$\mathsf{pd}_i \leftarrow \mathsf{PartDec}(\mathsf{ct}, \mathbf{s}_i, S_{\mathsf{act}}) := L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i + e_i \mod q. \text{ (naïve prototype)} \qquad (10)$$

Here, the active party set S_{act} must be known before threshold decryption; i.e., the parties need synchronization. Although [OT26a] simply assumes that S_{act} is somehow known, in practice, this requires one more round of communication for the parties to declare their participation and determine S_{act}. Afterwards, given $\{\mathsf{pd}_i\}_{i \in S_{\mathsf{act}}}$, the decryption is completed as follows:

$$\begin{aligned} \mathsf{FinDec}(\mathsf{ct}, \{\mathsf{pd}_i\}_{i \in S_{\mathsf{act}}}) &:= \lfloor (b - \textstyle\sum_{i \in S_{\mathsf{act}}} \mathsf{pd}_i \bmod q) / \lfloor \tfrac{q}{2} \rfloor \rceil \\ &= \mu + \lfloor (e_{\mathsf{ct}} - \textstyle\sum_{i \in S_{\mathsf{act}}} e_i) / \lfloor \tfrac{q}{2} \rfloor \rceil. \qquad (11) \end{aligned}$$

While the construction of (10) satisfies correctness, the security proof of this naïve construction is difficult; we cannot simulate pd_h as in (9) since the Lagrange coefficient is "embedded" in pd_i. Thus, [OT26a] uses a masking technique, inspired by the threshold signature schemes of [PKM+24]. More precisely, the decryption share of [OT26a] is in the following form:

$$\mathsf{pd}_i \leftarrow \mathsf{PartDec}(\mathsf{ct}, \mathbf{s}_i, S_{\mathsf{act}}) := L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i + \mathfrak{m}_i^{S_{\mathsf{act}}} \bmod q \ ([\text{OT26a, Algo. 3}]), \quad (12)$$

where $\mathfrak{m}_i^{S_{\mathsf{act}}}$ is a masking value produced by a pseudorandom function (PRF), and the seed values of $\mathfrak{m}_i^{S_{\mathsf{act}}}$ are shared with the semi-trusted server. The server adds $\sum_{i \in S_{\mathsf{act}}} \mathfrak{m}_i^{S_{\mathsf{act}}}$ to the b part of ct during $\mathsf{ServerDec}$; thus, FinDec correctly works as in (11), since the sum of masks is canceled. Owing to this masking technique, pd_i essentially becomes pseudorandom; thus, it can be simulated by a uniformly random value.

Our construction. Our second scheme is constructed by adapting the "synchronous and masked" Shamir-SS-based threshold decryption of [OT26a] to our first scheme in a distributed manner without the semi-trusted server. The decryption share of our scheme is constructed as follows;

$$\mathsf{pd}_i \leftarrow \mathsf{PartDec}(\mathsf{ct}, \mathbf{s}_i, S_{\mathsf{act}}) := \left\lfloor \tfrac{1}{p}(L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i \bmod Q) \right\rceil_\sigma + \Delta_i^{S_{\mathsf{act}}} \bmod q, \quad (13)$$

where $\Delta_i^{S_{\mathsf{act}}}$ is a *zero-share* masking of [KRT24], and $\lfloor \cdot \rceil_\sigma$ is the randomized Gaussian rounding (Def. 3.13). The mask $\Delta_i^{S_{\mathsf{act}}}$ is essentially pseudorandom except for the condition $\sum_{i \in S_{\mathsf{act}}} \Delta_i^{S_{\mathsf{act}}} = 0$. Although this technique is essentially equivalent to the masking technique of [PKM+24] used in [OT26a], the former can be performed in one round of communication; thus, it is compatible with threshold decryption without the server. The rest of the decryption procedure is almost identical to that of our first scheme described in Sect. 2.4.

Also note that, while partial decryption of [OT26a] in (12) does not contain an error term, our partial decryption in (13) contains the rounding error. Hence, although the overall proof strategy is similar to [OT26a], our proof requires a careful analysis of the randomized rounding errors, and as a result we need the sum of rounding errors to be simulatable. For this reason, the simple (deterministic) flooring ($\lfloor \cdot \rfloor$) as in the prior construction (6) is not sufficient for the Shamir-SS-based construction (13), and we require the randomized Gaussian rounding ($\lfloor \cdot \rceil_\sigma$).

3 Preliminaries

3.1 Notation and Basics

For a positive integer N, we denote $[N] := \{i \in \mathbb{N} \mid 1 \leq i \leq N\}$. The size of a set S is denoted by $|S|$. For $x \in \mathbb{R}$, we define $\{x\} := x - \lfloor x \rfloor$, the fractional part of x. For $q \in \mathbb{N}$, we define $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$. Whenever $\mathbb{Z}_q$ is embedded into $\mathbb{R}$, we identify it with its centered representative in $(-q/2, q/2]$. We use λ as a security parameter. Let $\mathrm{negl}(\lambda) = \lambda^{-\omega(1)}$ denote the class of negligible functions, and let $\mathrm{poly}(\lambda) = \lambda^{O(1)}$ denote the class of polynomial functions. We refer to functions of the form $1 - \mathrm{negl}(\lambda)$ as overwhelming. Bold lower-case letters denote vectors, and bold upper-case letters denote matrices. The transpose of $\mathbf{x}$ is written as $\mathbf{x}^\intercal$. We write $X \xleftarrow{\$} S$ when X is sampled uniformly at random from the set S. For two distributions χ_1, χ_2, we define $\chi_1 + \chi_2 := \{X_1 + X_2 \mid X_1 \leftarrow \chi_1, X_2 \leftarrow \chi_2\}$. We write $\chi_1 \approx \chi_2$ and $\chi_1 \approx_{\mathsf{s}} \chi_2$ when χ_1 and χ_2 are identically distributed and statistically indistinguishable, respectively. For $B > 0$, we say that a distribution χ over $\mathbb{R}$ is B-bounded if $\Pr_{X \leftarrow \chi}[|X| \geq B] = \mathrm{negl}(\lambda)$. The command $\mathbf{assert}\{\mathsf{condition}\}$ aborts and returns $\perp$ if $\mathsf{condition}$ is false, and otherwise does nothing. The command $\mathbf{pop}(\mathsf{List}, \mathsf{idx})$ outputs the idx-th element of the indexed set List, and updates List by removing the idx-th element.

We recall a technique called "smudging lemma" or "noise flooding":

Lemma 3.1 (Smudging Lemma [AJL+12; MW16]). *Let $B_1, B_2 \in \mathbb{N}$. For any $e_1 \in [-B_1, B_1]$, let E_1 and E_2 be independent random variables uniformly distributed over $[-B_2, B_2]$ and define the two random variables $X_1 = E_1 + e_1$ and $X_2 = E_2$. Then, $\Delta(X_1, X_2) < B_1/B_2$.*

3.2 Binary coefficient linear secret sharing ($\{0,1\}$-LSS)

In this subsection, we describe the construction of binary coefficient linear secret sharing ($\{0,1\}$-LSS) of [BGG+18]. We let $P = \{P_1, \ldots, P_N\}$ be a set of parties. The power set of a set S is $\mathbb{P}(S) = \{A \mid A \subseteq S\}$. A collection $\mathbb{A} \subseteq \mathbb{P}(P)$ is monotone if $B \in \mathbb{A}$ implies $C \in \mathbb{A}$ for any $B \subseteq C (\subseteq P)$, and a monotone access structure on P is a monotone collection $\mathbb{A} \subseteq \mathbb{P}(P) \setminus \varnothing$.

Definition 3.2 (Valid party set). *Let $\mathbb{A}$ be a monotone access structure on P. The sets $S \in \mathbb{A}$ are called the valid sets and the sets $S \in \mathbb{P}(P) \setminus \mathbb{A}$ are called the invalid sets. Furthermore, we define the following:*

- *Maximal invalid party set:* $\{S \notin \mathbb{A} \mid \forall P_i \in P \setminus S, S \cup \{P_i\} \in \mathbb{A}\}$
- *Minimal valid party set:* $\{S \in \mathbb{A} \mid \forall S' \subsetneq S, S' \notin \mathbb{A}\}$

We define $\{0,1\}$-LSS of [BGG+18] as follows:

Definition 3.3. *Let $\mathbb{S}$ be a class of efficient access structures on P. The binary coefficient linear secret sharing scheme over $\mathbb{Z}_q$ is a tuple of algorithms $\{0,1\}$-$\mathsf{LSS} := (\{0,1\}\text{-}\mathsf{LSS.Share}, \{0,1\}\text{-}\mathsf{LSS.Combine})$ defined as follows:*

- $\{0,1\}\text{-}\mathsf{LSS.Share}(\mathsf{k} \in \mathbb{Z}_q, \mathbb{A} \in \mathbb{S}) \to (\mathsf{s}_1, \ldots \mathsf{s}_N)$*: Set a share matrix $\mathbf{M} \in \mathbb{Z}_q^{l \times N}$ corresponding to the access structure $\mathbb{A}$, and associate each party P_i with an index set $I_i \subseteq [l]$. The sharing algorithm first samples $r_2, \ldots, r_N \xleftarrow{\$} \mathbb{Z}_q$ and calculates $\mathbf{w} := (w_1, \ldots, w_l)^\intercal$, defined as $\mathbf{w} = \mathbf{M} \cdot (\mathsf{k}, r_2, \ldots, r_N)^\intercal$. Then, $\mathsf{s}_i := \{w_j\}_{j \in I_i}$ is output as the share of the secret k for P_i.*
- $\{0,1\}\text{-}\mathsf{LSS.Combine}(\{\mathsf{s}_i\}_{i \in S}) \to \mathsf{k}$*: Any valid set of parties $S \in \mathbb{A}$ can efficiently find the binary coefficients $\{c_j \in \{0,1\}\}_{j \in \bigcup_{i \in S} I_i}$ s.t. $\sum_{j \in \bigcup_{i \in S} I_i} c_j \mathbf{M}[j] = (1, 0, \ldots, 0)$; i.e., a subset of indices $I \subseteq \bigcup_{i \in S} I_i$ s.t. $\mathsf{k} = \sum_{j \in I} w_j$; Then, output the secret $\mathsf{k} = \sum_{j \in I} w_j$.*

The correctness and privacy of $\{0,1\}$-LSS are defined as follows:

- *Correctness: For any valid party set $S \in \mathbb{A}$, $\mathsf{k} \in \mathbb{Z}_q$, $(\mathsf{s}_1, \ldots, \mathsf{s}_N) \leftarrow \{0,1\}$-$\mathsf{LSS.Share}(\mathsf{k}, \mathbb{A})$, we (always) obtain $\{0,1\}\text{-}\mathsf{LSS.Combine}(\{\mathsf{s}_i\}_{i \in S}) = \mathsf{k}$.*
- *Privacy: For any invalid party set $S \notin \mathbb{A}$, $\mathsf{k}_b \in \mathbb{Z}_q$, $(\mathsf{s}_{b,1}, \ldots, \mathsf{s}_{b,N}) \leftarrow \{0,1\}$-$\mathsf{LSS.Share}(\mathsf{k}_b, \mathbb{A})$ for $b \in \{0,1\}$, we have $\{\mathsf{s}_{0,i}\}_{i \in S} \approx \{\mathsf{s}_{1,i}\}_{i \in S}$.*

We define an analog of Def. 3.2 for the set of shares as follows:

Definition 3.4 (Valid share set). *Let $\mathbb{S}$ be a class of efficient access structures on P, and let SS be a $\{0,1\}$-LSS for $\mathbb{S}$ with share matrix $\mathbf{M} \in \mathbb{Z}_q^{l \times N}$. For a set of indices $I \subseteq [l]$, we say that I is a valid share set if binary coefficients $\{c_j \in \{0,1\}\}_{j \in I}$ s.t. $\sum_{j \in I} c_j \cdot \mathbf{M}[j] = (1, 0, \ldots, 0)$ exist. Otherwise,*

I is an invalid share set. We also define the maximal invalid share set as $\{$invalid $I \subseteq [l] \mid \forall i \in [l] \setminus I, (I \cup i)$ is valid$\}$, and define the minimal valid share set as $\{$valid $I \subseteq [l] \mid \forall I' \subsetneq I, I'$ is invalid$\}$.

$\{0,1\}$-LSS (Def. 3.3) corresponds to any threshold access structure:

Theorem 3.5 ([BGG+18, Theorem 4.15]). *There exists an efficient, correct and secure $\{0,1\}$-LSS scheme for any (T,N)-threshold access structure.*

Note that a crucial drawback of $\{0,1\}$-LSS is that it increases the key share sizes by a factor of $O(N^{4.3})$:

Lemma 3.6 ([Val84; Gol20; BGG+18]). *To represent an arbitrary (T,N)-threshold access structure via $\{0,1\}$-LSS defined as in Def. 3.3, any minimal valid share set I satisfies $|I| = O(N^{5.3})$, and the index set $I_i \subseteq [l]$ of each party P_i satisfies $|I_i| = O(N^{4.3})$.*

By performing the algorithm in an element-wise manner, $\{0,1\}$-LSS naturally extends to secrets in vector form.

3.3 Shamir secret sharing (Shamir-SS)

We define Shamir's secret sharing as follows:

Construction 3.7. *Let N be a number of parties, and let $\mathbb{K}$ be a ring (or a field) in which $i-j$ is invertible for all distinct $i,j \in [N]$. Shamir's secret sharing over $\mathbb{K}$ with threshold $T \leq N$ is a tuple of the algorithms $\mathsf{Shamir}_{\mathbb{K},T,N} := (\mathsf{Share}, \mathsf{Recon})$ defined as follows:*

- $\mathsf{Share}(s \in \mathbb{K})$*: Select a polynomial $P \in \mathbb{K}[X]$ of degree at most $T-1$ such that $P(0) = s$, where $\mathbb{K}[X]$ denotes the set of polynomials over $\mathbb{K}$. Output the vector of shares $(s_1, \ldots, s_N) := (P(1), \ldots, P(N)) \in \mathbb{K}^N$.*
- $\mathsf{Recon}(\{s_i\}_{i \in S \subseteq [N]})$*: Outputs $s = \sum_{i \in S} L_i^S \cdot s_i$, where $L_i^S := \prod_{j \in S \setminus \{i\}} \left(\frac{-j}{i-j}\right)$.*

Shamir-SS satisfies correctness and privacy as in Def. 3.3. By performing the algorithm in an element-wise manner, Shamir-SS naturally extends to secrets in vector form.

3.4 Lattices and Gaussians

A lattice with basis $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$ is defined as $\mathcal{L} = \{\sum_{i=1}^n z_i \mathbf{b}_i \mid \mathbf{z} \in \mathbb{Z}^n\}$, and the dual of a lattice $\mathcal{L}$ is defined as $\widehat{\mathcal{L}} := \{\mathbf{x} \mid \forall \mathbf{y} \in \mathcal{L}, \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}$. For a full rank matrix $\mathbf{S} \in \mathbb{R}^{n \times m}$ and a center $\mathbf{c} \in \mathbb{R}^n$, the Gaussian function on $\mathbb{R}^n$ is defined as $\rho_{\mathbf{S},\mathbf{c}}(\mathbf{x}) := \exp(-\pi(\mathbf{x}-\mathbf{c})^\intercal(\mathbf{S}\mathbf{S}^\intercal)^{-1}(\mathbf{x}-\mathbf{c}))$. For any set $A \subseteq \mathbb{R}^n$, we define $\rho_{\mathbf{S},\mathbf{c}}(A) := \sum_{\mathbf{x} \in A} \rho_{\mathbf{S},\mathbf{c}}(\mathbf{x})$. The *smoothing parameter* of $\mathcal{L}$ is defined as $\eta_\epsilon(\mathcal{L}) = \min\{s \mid \rho_{1/s}(\widehat{\mathcal{L}}) \leq 1+\epsilon\}$ for $\epsilon > 0$. The discrete Gaussian distribution over the lattice $\mathcal{L}$ is defined as follows:

Definition 3.8 (Discrete Gaussian). *For a full rank matrix* $\mathbf{S}$*, the discrete Gaussian distribution over a lattice* $\mathcal{L}$ *with a center* $\mathbf{c} \in \mathbb{R}^n$ *is defined as* $D_{\mathcal{L},\mathbf{S},\mathbf{c}}(\mathbf{x}) = \rho_{\mathbf{S},\mathbf{c}}(\mathbf{x})/\rho_{\mathbf{S},\mathbf{c}}(\mathcal{L})$*. If* $\mathbf{S}\mathbf{S}^\intercal = s^2\mathbf{I}_n$ *for* $s > 0$*, we write* $D_{\mathcal{L},\mathbf{S},\mathbf{c}}$ *as* $D_{\mathcal{L},s,\mathbf{c}}$*. The subscript* $\mathbf{c}$ *is omitted when* $\mathbf{c} = \mathbf{0}$*.*

A linear sum of discrete Gaussian variables is close to a discrete Gaussian:

Lemma 3.9 (Special case of [MP13, Theorem 3.3]). *Let* $\epsilon = \mathrm{negl}(\lambda)$*,* $\sigma \geq \sqrt{2}\eta_\epsilon(\mathbb{Z})$*, and* $e_i \leftarrow D_{\mathbb{Z}+c_i,\sigma}$ *for* $i \in [m]$*. Then, we have* $\sum_{i\in[m]} e_i \approx_{\mathsf{s}} \mathcal{D}_{\mathbb{Z}+\sum_{i\in[m]} c_i,\sqrt{m}\sigma}$*.*

A tail bound for the discrete Gaussian can be obtained as follows:

Lemma 3.10. *For any* $\epsilon \in (0,1)$*,* $s \geq \eta_\epsilon(\mathbb{Z})$ *and* $t \in \mathbb{N}$*, we obtain* $\Pr_{x\leftarrow D_{\mathbb{Z},s}}[|x| \geq t] \leq \frac{s}{(1-\epsilon)\pi t} e^{-\pi t^2/s^2}$*. In particular, for* $t = s \cdot \Omega(\sqrt{\lambda})$*, we obtain* $\Pr_{x\leftarrow D_{\mathbb{Z},s}}[|x| \geq t] = \mathrm{negl}(\lambda)$*, i.e.,* $D_{\mathbb{Z},s}$ *is* $s \cdot \Omega(\sqrt{\lambda})$*-bounded.*

Passelègue and Stehlé [PS25] presented the smudging lemma for Gaussian.

Lemma 3.11 ([PS25, Lemma 2.1]). *For any* $\sigma > 0$ *and* $c_0, c_1 \in \mathbb{R}$*, we have:*

$$\Delta(D_{\mathbb{Z},\sigma,c_0}, D_{\mathbb{Z},\sigma,c_1}) = O\left(\frac{|c_0 - c_1|}{\sigma}\right),$$

where $\Delta(\cdot,\cdot)$ *denotes the statistical distance. In particular, for* $\lambda > 0$*,* $c \in \mathbb{R}$ *and* $\sigma = \Omega(|c| \cdot 2^\lambda)$*, we have* $\Delta(D_{\mathbb{Z},\sigma}, D_{\mathbb{Z},\sigma,c}) = O(2^{-\lambda})$ *(and thus,* $D_{\mathbb{Z},\sigma} \approx_{\mathsf{s}} D_{\mathbb{Z},\sigma,c}$*).*

The above lemma is extended to a discrete Gaussian over any coset of integer lattices; $s + \mathbb{Z}$ for any shift $s \in \mathbb{R}$:

Corollary 3.12. *For any* $\sigma > 0$ *and* $c_0, c_1, s \in \mathbb{R}$*, we have:*

$$\Delta(D_{\mathbb{Z}+s,\sigma,c_0}, D_{\mathbb{Z}+s,\sigma,c_1}) = O\left(\frac{|c_0 - c_1|}{\sigma}\right).$$

Proof. The map $x \mapsto x - s$ is a bijection from $\mathbb{Z}+s$ to $\mathbb{Z}$ and transforms $D_{\mathbb{Z}+s,\sigma,c}$ into $D_{\mathbb{Z},\sigma,c-s}$. Hence, we have $\Delta(D_{\mathbb{Z}+s,\sigma,c_0}, D_{\mathbb{Z}+s,\sigma,c_1}) = \Delta(D_{\mathbb{Z},\sigma,c_0-s}, D_{\mathbb{Z},\sigma,c_1-s})$. Then, by Lem. 3.11, we obtain $\Delta(D_{\mathbb{Z},\sigma,c_0-s}, D_{\mathbb{Z},\sigma,c_1-s}) = O\left(\frac{|(c_0-s)-(c_1-s)|}{\sigma}\right)$. □

Finally, we define the randomized Gaussian rounding operation, which is of great importance in our second construction in Sect. 6 (and [PS25]). Given a real value $c \in \mathbb{R}$, this operation randomly outputs an integer point around the center c according to the discrete Gaussian distribution; i.e., it outputs $\mathcal{D}_{\mathbb{Z},\sigma,c}$, a discrete Gaussian with center c, which is illustrated in Fig. 3.

Definition 3.13 ($\lfloor\cdot\rceil_\sigma$). *On input* $c \in \mathbb{R}$ *and a parameter* $\sigma > 0$*, the randomized Gaussian rounding procedure outputs* $\lfloor c \rceil_\sigma \leftarrow D_{\mathbb{Z},\sigma,c}$*. (Equivalently, it samples* $z \leftarrow D_{\mathbb{Z}-\{c\},\sigma}$ *and returns* $c + z$*, where* $\{c\} = c - \lfloor c \rfloor$ *is the fractional part of* c*.)*

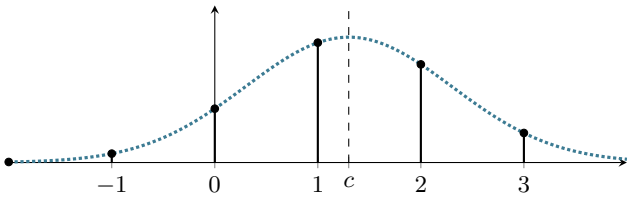


Fig. 3. Probability mass function of $\mathcal{D}_{\mathbb{Z},\sigma,c}$ with $c = 1.3$ and $\sigma = 1$

3.5 Learning with errors (LWE)

The LWE distribution and the LWE problem are defined as follows:

Definition 3.14 (LWE). *Let n, m, and q be integers and χ be an error distribution over $\mathbb{Z}_q$. The LWE distribution is defined as*

$$\mathsf{LWE}(n, m, q, \chi) := \left\{ (\mathbf{A}, \mathbf{b} := \mathbf{As} + \mathbf{e}) \mid \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}, \mathbf{s} \leftarrow \chi^n, \mathbf{e} \leftarrow \chi^m \right\}.$$

The advantage of an algorithm $\mathcal{A}$ for solving the (decision) LWE problem is defined as

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{LWE}} = |\Pr[\mathcal{A}(\mathbf{A}, \mathbf{As} + \mathbf{e}) = 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^m) = 1]|.$$

3.6 Pseudorandom function

We rely on the multi-instance variant of pseudorandom functions (PRFs) as provided in [PKM+24]:

Definition 3.15 (Pseudorandom function). *Let λ be a security parameter and $n, l \in \mathbb{N}$. We say that a deterministic PPT algorithm $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^l$ is a pseudorandom function if for any PPT algorithm $\mathcal{A}$*

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{PRF}}(\lambda) := |\Pr[\mathsf{Game}_{\mathcal{A}}^{\mathsf{PRF}}(\lambda) = 1] - \tfrac{1}{2}| = \mathrm{negl}(\lambda)$$

holds, where $\mathsf{Game}_{\mathcal{A}}^{\mathsf{PRF}}$ is defined as in Algo. 1.

For simplicity, we write $\mathsf{PRF}(\mathsf{seed}, x) \in S$, assuming a fixed canonical encoding of the input x and of PRF outputs into S. We encode variable-length inputs using an injective, length-delimited, domain-separated canonical encoding.

4 Definitions of FHE and ThFHE

We define the syntax, correctness and security of FHE and ThFHE. We also provide the framework of the underlying FHE used in this paper.

Algorithm 1: Security game of PRF.

$\underline{\mathsf{Game}^{\mathsf{PRF}}_{\mathcal{A}}(\lambda)}$:
$\mathsf{List}_{\mathsf{seed}}[\cdot] := \emptyset$
$\mathsf{List}_{\mathsf{out}}[\cdot,\cdot] := \emptyset$
$b \leftarrow \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{PRF}}}(\lambda)$
if $b = b'$ **then return** 1
else return 0

$\underline{\mathcal{O}_{\mathsf{PRF}}(\mathsf{idx}, x \in \{0,1\}^n)}$:
if $\mathsf{List}_{\mathsf{seed}}[\mathsf{idx}] = \emptyset$ **then**
  $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$,
  $\mathsf{List}_{\mathsf{seed}}[\mathsf{idx}] \leftarrow \mathsf{seed}$
$\mathsf{seed} := \mathsf{List}_{\mathsf{seed}}[\mathsf{idx}]$
if $\mathsf{List}_{\mathsf{out}}[\mathsf{idx}, x] = \emptyset$ **then**
  $y_0 \xleftarrow{\$} \{0,1\}^l$, $\mathsf{List}_{\mathsf{out}}[\mathsf{idx}, x] \leftarrow y_0$
else
  $y_0 := \mathsf{List}_{\mathsf{out}}[\mathsf{idx}, x]$
$y_1 \leftarrow \mathsf{PRF}(\mathsf{seed}, x)$
return y_b

4.1 Syntax and structure of the underlying FHE

The syntax of FHE is defined as follows:

Definition 4.1 (FHE). *A fully homomorphic encryption (FHE) for a plaintext space $\mathcal{M}$ and a ciphertext space $\mathcal{C}$ is a tuple of algorithms* $\mathsf{FHE} = (\mathsf{PP}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *defined as follows;*

- $\mathsf{FHE.PP}(1^\lambda) \to \mathsf{pp}$*: On input a security parameter λ, outputs a set of public parameters* pp*. The subsequent algorithms implicitly take* pp *as an argument.*
- $\mathsf{FHE.KeyGen}() \to (\mathsf{evk}, \mathsf{pk}, \mathsf{sk})$*: Outputs a public evaluation key* evk*, a public key* pk *and a secret key* sk*.*
- $\mathsf{FHE.Enc}(\mathsf{pk}, \mu \in \mathcal{M}) \to \mathsf{ct} \in \mathcal{C}$ *and* $\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{ct} \in \mathcal{C}) \to \overline{\mu} \in \mathcal{M}$*: Have the standard syntax for public-key encryption/decryption.*
- $\mathsf{FHE.Eval}(\mathsf{evk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_l \in \mathcal{C}) \to \mathsf{ct}_{\mathsf{eval}} \in \mathcal{C}$*: On input of the evaluation key* evk*, a function $C : \mathcal{M}^l \to \mathcal{M}$, and a set of l ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_l$ of $\mu_1, \ldots, \mu_l$, outputs the result ciphertext* $\mathsf{ct}_{\mathsf{eval}}$*.* FHE *is* correct*, if* $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_{\mathsf{eval}}) = C(\mu_1, \ldots, \mu_l)$ *holds with overwhelming probability.*

We define in Algo. 2 a structure of the underlying FHE scheme based on LWE, which is identical to that of [PS25]. This scheme is based on the standard LWE-based PKE of [LP11], and its IND-CPA security can be shown under the hardness assumption of $\mathsf{LWE}(n, m, Q, \chi_{\mathsf{lwe}})$. We also assume that the underlying FHE is correct; the ciphertext noise e_{ct} (even after Eval) is assumed to satisfy $|e_{\overline{\mathsf{ct}}}| < B_{\mathsf{err}}$ for $B_{\mathsf{err}} < \frac{Q}{4}$. Our ThFHE scheme is constructed based on this underlying FHE scheme (Algo. 2). This framework subsumes most known FHE schemes, although it does not extend to GSW-like schemes [GSW13]. We can naturally extend this FHE scheme to MLWE-based construction [BGV12].

4.2 Syntax of our threshold FHE

The syntax of our ThFHE is slightly generalized from the standard one, e.g., [BGG+18]. Our partial decryption $\mathsf{PartDec}$ takes an additional argument

Algorithm 2: Underlying $\mathsf{FHE} := (\mathsf{PP}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$

$\underline{\mathsf{PP}(1^\lambda) \to \mathsf{pp}}$:
return $\mathsf{pp} := (n, m, Q, \chi_{\mathsf{lwe}}, B_{\mathsf{err}})$

$\underline{\mathsf{KeyGen}()}$:
Sample $\mathsf{sk} := \mathbf{s} \leftarrow \chi_{\mathsf{lwe}}^n$ and $\mathsf{pk} := (\mathbf{A}, \mathbf{b} := \mathbf{As} + \mathbf{e}) \leftarrow \mathsf{LWE}(n, m, Q, \chi_{\mathsf{lwe}})$
Generate the evaluation key evk ▷ Details of evk are omitted
return $(\mathsf{evk}, \mathsf{pk}, \mathsf{sk})$

$\underline{\mathsf{Enc}(\mathsf{pk}, \mu \in \mathcal{M} := \{0,1\})}$:
Sample $\mathbf{r} \leftarrow \chi_{\mathsf{lwe}}^m, \mathbf{f} \leftarrow \chi_{\mathsf{lwe}}^n$ and $f \leftarrow \chi_{\mathsf{lwe}}$
return $\mathsf{ct} := (\mathbf{a}, b) := (\mathbf{r}^\intercal \mathbf{A} + \mathbf{f}, \mathbf{r}^\intercal \mathbf{b} + f + \lfloor \frac{Q}{2} \rfloor \cdot \mu)$
We assume the ciphertext noise $e_{\mathsf{ct}} := b - \mathbf{a}^\intercal \mathbf{s} - \lfloor \frac{Q}{2} \rfloor \cdot \mu$ satisfies $|e_{\mathsf{ct}}| < B_{\mathsf{err}}$.

$\underline{\mathsf{Eval}(\mathsf{evk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_l)}$: ▷ We denote plaintexts of $\mathsf{ct}_1, \ldots, \mathsf{ct}_l$ by $\mu_1, \ldots, \mu_l$
return $\overline{\mathsf{ct}} := (\overline{\mathbf{a}}, \overline{b})$
We assume $e_{\overline{\mathsf{ct}}} := \overline{b} - \overline{\mathbf{a}}^\intercal \mathbf{s} - \lfloor \frac{Q}{2} \rfloor \cdot C(\mu_1, \ldots, \mu_l)$ satisfies $|e_{\overline{\mathsf{ct}}}| < B_{\mathsf{err}}$ for any C.

$\underline{\mathsf{Dec}(\mathsf{sk}, \mathsf{ct} := (\mathbf{a}, b)) \to \overline{\mu} \in \mathcal{M}}$:
return $\overline{\mu} := \lfloor (b - \mathbf{a}^\intercal \mathbf{s}) / \lfloor \frac{Q}{2} \rfloor \rceil$

$S_{\mathsf{act}} \subseteq [N]$, which is an *active party set* of the threshold decryption session. Note that although $\mathsf{PartDec}$ of our ThFHE from Shamir-SS (Sect. 6) uses S_{act}, $\mathsf{PartDec}$ of our ThFHE from $\{0,1\}$-LSS (Sect. 5) does not use sid and S_{act}.

Definition 4.2 (Threshold FHE). *Let $P_1, \ldots, P_N$ be parties and let $T \leq N$ be a threshold. A threshold FHE scheme for a plaintext space $\mathcal{M}$, a ciphertext space $\mathcal{C}$ and a decryption share space $\mathcal{C}_{\mathsf{pd}}$ is a tuple of algorithms $\mathsf{ThFHE} = (\mathsf{PP}, \mathsf{KeyGen}, \mathsf{ShareGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$ defined as follows:*

- $\mathsf{ThFHE.PP}(1^\lambda, 1^N) \to \mathsf{pp}$*: On input a security parameter λ and the number of parties N, outputs a set of public parameters pp. The subsequent algorithms implicitly take pp as an input.*
- $\mathsf{ThFHE.KeyGen}() \to (\mathsf{evk}, \mathsf{pk}, \mathsf{sk})$*: Same as* $\mathsf{FHE.KeyGen}$ *in Def. 4.1.*
- $\mathsf{ThFHE.ShareGen}(\mathsf{sk}, T, N) \to (\{\mathsf{sk}_i\}_{i \in [N]})$*: On input of the secret key sk, T and N, outputs secret shares $\{\mathsf{sk}_i\}_{i \in [N]}$.*
- $\mathsf{ThFHE.Enc}(\mathsf{pk}, \mu \in \mathcal{M}) \to \mathsf{ct} \in \mathcal{C}$*: Same as* $\mathsf{FHE.Enc}$ *in Def. 4.1.*
- $\mathsf{ThFHE.Eval}(\mathsf{evk}, C, \mathsf{ct}_1, \ldots \mathsf{ct}_l) \to \mathsf{ct}_{\mathsf{eval}} \in \mathcal{C}$*: Same as* $\mathsf{FHE.Eval}$ *in Def. 4.1.*
- $\mathsf{ThFHE.PartDec}(\mathsf{ct}, \mathsf{sk}_i, \mathsf{sid}, S_{\mathsf{act}}) \to \mathsf{pd}_i$*: On input of a ciphertext ct, a secret share sk_i, a session ID $\mathsf{sid} \in \mathbb{N}$, and a set of active parties S_{act}, outputs a partial decryption share pd_i. (*sid *and* S_{act} *are omitted in the construction in Sect. 5.)*
- $\mathsf{ThFHE.FinDec}(\mathsf{ct}, \{\mathsf{pd}_i\}_{i \in S}) \to \mu$*: On input of a ciphertext ct and a set of partial decryption shares $\{\mathsf{pd}_i\}_{i \in S}$ for some $S \subseteq [N]$, outputs a decryption result $\mu \in \mathcal{M} \cup \{\perp\}$.*

We define the correctness of ThFHE as follows:

Algorithm 3: Correctness game for our ThFHE

$\underline{\mathsf{Game}^{\mathsf{correct}}_{\mathsf{ThFHE}}(1^\lambda, T, N, \mathsf{sid}, l, C, (\mu_1, \ldots, \mu_l) \in \mathcal{M}^l)}$:

$\mathsf{pp} \leftarrow \mathsf{PP}(1^\lambda, 1^N)$, $(\mathsf{evk}, \mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$, $\{\mathsf{sk}_i\}_{i\in[N]} \leftarrow \mathsf{ShareGen}(\mathsf{sk}, T, N)$
for $j \in [l]$ **do** $\mathsf{ct}_j \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu_j)$
$\mathsf{ct} \leftarrow \mathsf{Eval}(\mathsf{evk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_l)$, and randomly choose $S_{\mathsf{act}} \subseteq [N]$ s.t. $|S_{\mathsf{act}}| \geq T$
for $i \in S_{\mathsf{act}}$ **do** $\mathsf{pd}_i \leftarrow \mathsf{PartDec}(\mathsf{ct}, \mathsf{sk}_i, \mathsf{sid}, S_{\mathsf{act}})$
$\overline{\mu} \leftarrow \mathsf{FinDec}(\mathsf{ct}, \{\mathsf{pd}_i\}_{i\in S_{\mathsf{act}}})$
if $\overline{\mu} = C(\mu_1, \ldots, \mu_l)$ **then return** 1 **else return** 0

Algorithm 4: The real experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$ and the ideal experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}$ for our security definition (Def. 4.4). The differences between $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$ and $\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}$ are highlighted.

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

$\mathsf{pp} \leftarrow \mathsf{PP}(1^\lambda, 1^N)$, $(\mathsf{evk}, \mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$,
$\{\mathsf{sk}_i\}_{i\in[N]} \leftarrow \mathsf{ShareGen}(\mathsf{sk}, T, N)$
$(S_{\mathsf{corr}} \subset [N]$ s.t. $|S_{\mathsf{corr}}| = T-1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{pk})$ ▷ S_{corr}: maximal invalid set
$S_{\mathsf{hon}} := [N] \setminus S_{\mathsf{corr}}$, $\mathsf{List} \leftarrow \emptyset$, $\mathsf{sid} := 0$
return $\{0,1\} \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{Eval}}, \mathcal{O}_{\mathsf{Dec}}}(\{\mathsf{sk}_i\}_{i\in S_{\mathsf{corr}}}, \mathsf{st})$

$\underline{\mathcal{O}_{\mathsf{Enc}}(\mu)}$:

$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu)$, $\mathsf{List} \leftarrow \mathsf{List} \cup (\mu, \mathsf{ct})$

$\underline{\mathcal{O}_{\mathsf{Eval}}(C, (\mathsf{idx}_1, \ldots, \mathsf{idx}_l))}$:

assert $\{\mathsf{idx}_1, \ldots, \mathsf{idx}_l \leq |\mathsf{List}|\}$
for $j \in [l]$ **do** $(\mu_j, \mathsf{ct}_j) \leftarrow \mathsf{List}[\mathsf{idx}_j]$
$\mathsf{ct} \leftarrow \mathsf{Eval}(\mathsf{evk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_l)$, $\mathsf{List} \leftarrow \mathsf{List} \cup (C(\mu_1, \ldots, \mu_l), \mathsf{ct})$

$\underline{\mathcal{O}_{\mathsf{Dec}}(\mathsf{idx}, S_{\mathsf{act}})}$:

assert $\{|S_{\mathsf{act}}| \geq T$ and $\mathsf{idx} \leq |\mathsf{List}|\}$, $\mathsf{sid} \leftarrow \mathsf{sid} + 1$
$(\mu, \mathsf{ct}) \leftarrow \mathbf{pop}(\mathsf{List}, \mathsf{idx})$ ▷ Dec is executed only once for the same ct
for $i \in S_{\mathsf{act}}$ **do** $\mathsf{pd}_i \leftarrow \mathsf{PartDec}(\mathsf{ct}, \mathsf{sk}_i, \mathsf{sid}, S_{\mathsf{act}})$
return $(\mathsf{ct}, \{\mathsf{pd}_i\}_{i\in S_{\mathsf{act}}})$

$\underline{\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda)}$:

Same as $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}(1^\lambda)$ except that $\{\mathsf{sk}_i\}_{i\in[N]}$ is replaced with:
$\{\mathsf{sk}_i\}_{i\in[N]} \leftarrow \mathsf{ShareGen}(\mathbf{0}, T, N)$ ▷ Shares that are irrelevant to sk
▷ $\mathcal{O}_{\mathsf{Enc}}$ and $\mathcal{O}_{\mathsf{Eval}}$ are identical to those in $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$.

$\underline{\mathcal{O}_{\mathsf{Dec}}(\mathsf{idx}, S_{\mathsf{act}})}$: ▷ $\mathcal{A}$ can query any (valid) subset of parties $S_{\mathsf{act}} \subseteq [N]$

assert $\{|S_{\mathsf{act}}| \geq T$ and $\mathsf{idx} \leq |\mathsf{List}|\}$, $\mathsf{sid} \leftarrow \mathsf{sid} + 1$
$(\mu, \mathsf{ct}) \leftarrow \mathbf{pop}(\mathsf{List}, \mathsf{idx})$
$\{\mathsf{pd}_i\}_{i\in S_{\mathsf{act}}} \leftarrow \mathsf{Sim}(\mu, \mathsf{ct}, \{\mathsf{sk}_i\}_{i\in S_{\mathsf{corr}}}, \mathsf{sid}, S_{\mathsf{act}})$
return $(\mathsf{ct}, \{\mathsf{pd}_i\}_{i\in S_{\mathsf{act}}})$

Definition 4.3 (Correctness). *We define* $\mathsf{Game}^{\mathsf{correct}}_{\mathsf{ThFHE}}$ *in Algo. 3.* ThFHE *is correct if, for any* $\lambda, N, T \in [1, N]$, $\mathsf{sid}, l > 0$, $C : \mathcal{M}^l \to \mathcal{M}$, $(\mu_1, \ldots, \mu_l) \in \mathcal{M}^l$,

$$\Pr[\mathsf{Game}^{\mathsf{correct}}_{\mathsf{ThFHE}}(1^\lambda, T, N, \mathsf{sid}, l, C, (\mu_1, \ldots, \mu_l)) = 1] \geq 1 - \mathrm{negl}(\lambda).$$

Algorithm 5: Our ThFHE based on $\{0,1\}$-LSS; $\mathsf{ThFHE} := (\mathsf{PP}, \mathsf{KeyGen}, \mathsf{ShareGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$.

$\underline{\mathsf{PP}(1^\lambda, 1^N)}$:
return $\mathsf{pp} := (n, m, p, q, Q = p \cdot q, \chi_{\mathsf{lwe}}, B_{\mathsf{flood}}, B_{\mathsf{err}})$
▷ We let the following functions implicitly include pp as an argument

$\underline{\mathsf{KeyGen}()}$:
return $(\mathsf{evk}, \mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}()$ (Algo. 2)

$\underline{\mathsf{ShareGen}(\mathsf{sk}, T, N)}$:
$\{\mathsf{sk}_i := \{\mathbf{s}_j \in \mathbb{Z}_Q^n\}_{j \in I_i}\}_{i \in [N]} \leftarrow \{0,1\}\text{-}\mathsf{LSS}_{\mathbb{Z}_Q^n}(\mathsf{sk}, T, N)$ (Def. 3.3)
return $\{\mathsf{sk}_i\}_{i \in [N]}$

$\underline{\mathsf{Enc}(\mathsf{pk}, \mu \in \mathcal{M} := \{0,1\})}$:
return $\mathsf{ct} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, \mu)$ (Algo. 2)

$\underline{\mathsf{Eval}(\mathsf{evk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_l)}$:
return $\mathsf{ct}_{\mathsf{eval}} \leftarrow \mathsf{FHE.Eval}(\mathsf{evk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_l)$ (Algo. 2)

$\underline{\mathsf{PartDec}(\mathsf{ct} := (\mathbf{a}, b), \mathsf{sk}_i)}$: ▷ The argument sid and S_{act} are not used here
for $j \in I_i$ **do** $\mathsf{p}_j := \lfloor \frac{1}{p}(\mathbf{a}^\intercal \mathbf{s}_j + E_j \bmod Q) \rfloor \bmod q$, where $E_j \xleftarrow{\$} [-B_{\mathsf{flood}}, B_{\mathsf{flood}}]$
return $\mathsf{pd}_i := \{\mathsf{p}_j\}_{j \in I_i}$

$\underline{\mathsf{FinDec}(\mathsf{ct} := (\mathbf{a}, b), \{\mathsf{pd}_i\}_{i \in S_{\mathsf{act}}})}$:
assert $\{|S_{\mathsf{act}}| \geq T\}$
For all $i \in S_{\mathsf{act}}$, parse $\mathsf{pd}_i := \{\mathsf{p}_j\}_{j \in I_i}$
Calculate a minimal valid share set $I \subseteq \bigcup_{i \in S_{\mathsf{act}}} I_i$ (Def. 3.4) ▷ $\sum_{j \in I} \mathbf{s}_j = \mathsf{sk}$
return $\overline{\mu} := \lfloor (\lfloor \frac{b}{p} \rfloor - \sum_{j \in I} \mathsf{p}_j \bmod q) / \lfloor \frac{q}{2} \rfloor \rceil$

We define the simulation-based security of our ThFHE scheme in Def. 4.4.

Definition 4.4 (Security). ThFHE *is simulation-secure if, for any PPT adversary* $\mathcal{A}$*, there exists a PPT simulator* Sim *s.t.* $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{ThFHE}}(\lambda) := |\Pr[\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}(1^\lambda) = 1] - \Pr[\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda) = 1]| = \mathrm{negl}(\lambda)$*, where* $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}(1^\lambda)$ *and* $\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda)$ *are defined in Algo. 4.*

5 Our Threshold FHE based on $\{0,1\}$-LSS

In this section, we present our serverless ThFHE with short decryption shares based on $\{0,1\}$-LSS. We begin by presenting our construction and the threshold decryption procedure in Sect. 5.1. Subsequently, we prove the correctness in Sect. 5.2 and the security in Sect. 5.3.

5.1 Construction

We present our $\{0,1\}$-LSS-based ThFHE scheme in Algo. 5. While the construction is essentially based on [BGG+18, Construction 5.6], PartDec and FinDec

differ. In $\mathsf{PartDec}$ (line 7), the components of pd_i are rounded off to be short values over a polynomially small modulus q. In FinDec (line 12), the parties locally round off the b part of ct and complete decryption.

Our one-round threshold decryption procedure is described as follows.

Input/output. Let ct be an input ciphertext of the threshold decryption process. As an output of threshold decryption, all parties recover the plaintext μ of ct.

1st round. Each party $P_i \in \{P_1, \ldots, P_N\}$ performs $\mathsf{PartDec}$ on ct and generates a polynomially short partial decryption share $\mathsf{pd}_i := \{\mathsf{p}_j\}_{j \in I_i} \in \mathbb{Z}_q^{|I_i|}$, and then broadcasts pd_i to all the parties.

End of the 1st round. Assume that the first round is performed by the active parties $S_{\mathsf{act}} \subseteq [N]$ s.t. $|S_{\mathsf{act}}| \geq T$ and that all of the parties received $\{\mathsf{pd}_i\}_{i \in S_{\mathsf{act}}}$. Since S_{act} is a valid party set (Def. 3.2), all of the parties can successfully perform FinDec and obtain the plaintext μ under the condition of correctness (Thm. 5.1).

5.2 Correctness

We prove the correctness (Def. 4.3) of Algo. 5:

Theorem 5.1. *Let $p > B_{\mathsf{err}}$ and $q = \Omega((\frac{B_{\mathsf{flood}}}{p}+1)N^{5.3})$. Then, ThFHE (Algo. 5) is correct (Def. 4.3) if the underlying FHE (Algo. 2) is correct (Def. 4.1).*

Proof. Our goal is to analyze the upper bound of the noise revealed after FinDec on line 12, i.e., $\lfloor \frac{b}{p} \rfloor - \sum_{j \in I} \mathsf{p}_j$. We first rewrite p_j on line 7 as follows:

$$\begin{aligned} \mathsf{p}_j &:= \tfrac{1}{p}\mathbf{a}^\intercal \mathbf{s}_j + \tfrac{1}{p}E_j - r_j \bmod q, \text{ where } r_j := \{\tfrac{1}{p}(\mathbf{a}^\intercal \mathbf{s}_j + E_j \bmod Q)\} \\ &:= \tfrac{1}{p}\mathbf{a}^\intercal \mathbf{s}_j + e_j \bmod q, \text{ where } e_j := \tfrac{1}{p}E_j - r_j. \end{aligned} \tag{14}$$

(Recall that $\{x\} := x - \lfloor x \rfloor$ is the fractional part.) Since I is valid, we obtain

$$\textstyle\sum_{j \in I} \mathsf{p}_j \bmod q = \sum_{j \in I}(\frac{1}{p}\mathbf{a}^\intercal \mathbf{s}_j + e_j) \bmod q = \frac{1}{p}\mathbf{a}^\intercal \mathbf{s} + \sum_{j \in I} e_j \bmod q. \tag{15}$$

Next, we analyze $\lfloor \frac{b}{p} \rfloor$ on line 12. The input ciphertext is denoted as $\mathsf{ct} := (\mathbf{a} \in \mathbb{Z}_Q^n, b := \mathbf{a}^\intercal \mathbf{s} + e_{\mathsf{ct}} + \lfloor \frac{Q}{2} \rfloor \cdot \mu \bmod Q)$, where $|e_{\mathsf{ct}}| < B_{\mathsf{err}}$ (cf. Algo. 2). We obtain

$$\begin{aligned} \lfloor \tfrac{b}{p} \rfloor \bmod q &= \tfrac{1}{p}(\mathbf{a}^\intercal \mathbf{s} + e_{\mathsf{ct}} + \lfloor \tfrac{Q}{2} \rfloor \cdot \mu \bmod Q) - \{\tfrac{b}{p}\} \bmod q \\ &= \tfrac{1}{p}\mathbf{a}^\intercal \mathbf{s} + \lfloor \tfrac{q}{2} \rfloor \cdot \mu + e' \bmod q, \end{aligned} \tag{16}$$

where we defined $e' := \frac{1}{p}e_{\mathsf{ct}} + (\{\frac{q}{2}\} - \frac{1}{p}\{\frac{Q}{2}\}) \cdot \mu - \{\frac{b}{p}\}$, and we used the fact that $\frac{1}{p}\lfloor \frac{Q}{2} \rfloor = \lfloor \frac{q}{2} \rfloor + \{\frac{q}{2}\} - \frac{1}{p}\{\frac{Q}{2}\}$. Finally, according to (15) and (16), we obtain

$$\textstyle\lfloor \frac{b}{p} \rfloor - \sum_{j \in I} \mathsf{p}_j \bmod q = \lfloor \frac{q}{2} \rfloor \cdot \mu + e' - \sum_{j \in I} e_j \bmod q. \tag{17}$$

We now derive a bound on the error term. By $|e_{\mathsf{ct}}| < B_{\mathsf{err}} < p$, we have

$$|e'| < \tfrac{1}{p} B_{\mathsf{err}} + (\tfrac{1}{2} + \tfrac{1}{2p}) \cdot \mu + 1 < 3. \tag{18}$$

From (14) and $|E_j| < B_{\mathsf{flood}}$, we obtain

$$|\textstyle\sum_{j \in I} e_j| < |I| \cdot (\frac{B_{\mathsf{flood}}}{p} + 1). \tag{19}$$

Note that we have $|I| = O(N^{5.3})$ by Lem. 3.6. Hence, by (17)–(19), we have

$$\lfloor (\lfloor \tfrac{b}{p} \rfloor - \textstyle\sum_{j \in I} \mathsf{p}_j) / \lfloor \tfrac{q}{2} \rfloor \rceil = \mu$$

for $q = \Omega((\frac{B_{\mathsf{flood}}}{p} + 1) N^{5.3})$. □

5.3 Security

We prove the security of Algo. 5 as follows:

Theorem 5.2. *Assume that p, q, B_{err} and B_{flood} are selected as in Thm. 5.1, and let $B_{\mathsf{flood}} = \Omega(2^\lambda B_{\mathsf{err}})$. Then,* ThFHE *(Algo. 5) is simulation-secure (Def. 4.4).*

Proof. We define a hybrid experiment Hyb in Algo. 6 and prove that $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}} \approx_{\mathsf{s}} \mathsf{Hyb} \approx \mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}$ to obtain the claim, where $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$ and $\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}$ are defined as in Algo. 4. In Hyb, the challenger first calculates and fixes a maximal invalid share set (Def. 3.4) $I^* \subseteq \bigcup_{i \in S_{\mathsf{corr}}} I_i$ on line 5. Since the corrupted party set S_{corr} is a maximal invalid party set (Def. 3.2), the challenger can efficiently calculate such I^*. Then, in $\mathcal{O}_{\mathsf{Dec}}$ of Hyb, the challenger simulates the partial decryption shares of active parties $\{\mathsf{pd}_i := \{\mathsf{p}_j\}_{j \in I_i}\}_{i \in S_{\mathsf{act}}}$ by using $\{\mathbf{s}_j\}_{j \in I^*}$. For $j \in I_i \cap I^*$, the challenger calculates p_j in the same way as that in $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$ (line 10). For $j \in I_i \setminus I^*$, the challenger first calculates a minimal valid share set $I \subseteq I^* \cup \{j\}$. Since I^* is a maximal invalid share set, the challenger can compute such I. Then, the challenger simulates p_j by using $\{\mathbf{s}_k\}_{k \in I \setminus \{j\}}$ (note that $I \setminus \{j\} \subseteq I^*$) as on line 14.

First, we show that $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}} \approx_{\mathsf{s}} \mathsf{Hyb}$ holds. We analyze p_j on line 14 of Hyb. The input ciphertext is denoted by $\mathsf{ct} := (\mathbf{a} \in \mathbb{Z}_Q^n, b := \mathbf{a}^\intercal \mathbf{s} + e_{\mathsf{ct}} + \lfloor \frac{Q}{2} \rfloor \cdot \mu \bmod Q)$. Then, for p_j on line 14, we have

$$\mathsf{p}_j = \lfloor \tfrac{1}{p} (\mathbf{a}^\intercal \mathbf{s}_j + e_{\mathsf{ct}} + E_j \bmod Q) \rfloor. \tag{20}$$

Here, we use the fact that $\sum_{k \in I \setminus \{j\}} \mathbf{s}_k = \mathbf{s} - \mathbf{s}_j$ by the correctness of $\{0,1\}$-LSS. Furthermore, by the smudging lemma (Lem. 3.1) and the hypothesis of $B_{\mathsf{flood}} = \Omega(2^\lambda B_{\mathsf{err}})$, $|e_{\mathsf{ct}}| < B_{\mathsf{err}}$, and $E_j \xleftarrow{\$} [-B_{\mathsf{flood}}, B_{\mathsf{flood}}]$, we have

$$e_{\mathsf{ct}} + E_j \approx_{\mathsf{s}} E_j.$$

Thus, by (20), for p_j on line 14 of Hyb, we have

$$\mathsf{p}_j \approx_{\mathsf{s}} \lfloor \tfrac{1}{p} (\mathbf{a}^\intercal \mathbf{s}_j + E_j \bmod Q) \rfloor.$$

Algorithm 6: Hybrid experiment Hyb for the proof of Thm. 5.2.

$\underline{\mathsf{Hyb}}$:

1 $\mathsf{pp} \leftarrow \mathsf{PP}(1^\lambda, 1^N)$, $(\mathsf{evk}, \mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$,
2 $\{\mathsf{sk}_i\}_{i\in[N]} \leftarrow \mathsf{ShareGen}(\mathsf{sk}, T, N)$
3 $(S_{\mathsf{corr}} \subset [N] \text{ s.t. } |S_{\mathsf{corr}}| = T-1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{pk})$ ▷ S_{corr}: maximal invalid set
4 $S_{\mathsf{hon}} := [N] \setminus S_{\mathsf{corr}}$, $\mathsf{List} \leftarrow \emptyset$, $\mathsf{sid} := 0$
5 Calculate and fix a maximal invalid share set $I^* \subseteq \bigcup_{i\in S_{\mathsf{corr}}} I_i$.
6 **return** $\{0,1\} \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{Eval}}, \mathcal{O}_{\mathsf{Dec}}}(\{\mathsf{sk}_i\}_{i\in S_{\mathsf{corr}}}, \mathsf{st})$
▷ $\mathcal{O}_{\mathsf{Enc}}$ and $\mathcal{O}_{\mathsf{Eval}}$ are the same as those in $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$.

$\underline{\mathcal{O}_{\mathsf{Dec}}(\mathsf{idx}, S_{\mathsf{act}})}$:

7 **assert** $\{|S_{\mathsf{act}}| \geq T \text{ and } \mathsf{idx} \leq |\mathsf{List}|\}$, $\mathsf{sid} \leftarrow \mathsf{sid}+1$
8 $(\mu, \mathsf{ct} := (\mathbf{a}, b)) \leftarrow \mathbf{pop}(\mathsf{List}, \mathsf{idx})$
9 **for** $i \in S_{\mathsf{act}}$ **do**
10 **for** $j \in I_i \cap I^*$ **do** $\mathsf{p}_j \leftarrow \lfloor \frac{1}{p}(\mathbf{a}^\intercal \mathbf{s}_j + E_j \bmod Q) \rfloor \bmod q$ ▷ as in $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$
11 **for** $j \in I_i \setminus I^*$ **do**
12 Calculate a minimal valid share set $I \subseteq I^* \cup \{j\}$
13 $E_j \xleftarrow{\$} [-B_{\mathsf{flood}}, B_{\mathsf{flood}}]$
14 $\mathsf{p}_j \leftarrow \lfloor \frac{1}{p}(b - (\sum_{k\in I\setminus\{j\}} \mathbf{a}^\intercal \mathbf{s}_k) + E_j - \lfloor \frac{Q}{2} \rfloor \cdot \mu \bmod Q) \rfloor \bmod q$
15 $\mathsf{pd}_i := \{\mathsf{p}_j\}_{j\in I_i}$
16 **return** $(\mathsf{ct}, \{\mathsf{pd}_i\}_{i\in S_{\mathsf{act}}})$

Since the right-hand side is equal to p_j in $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$, we obtain $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}} \approx_{\mathsf{s}} \mathsf{Hyb}$.

Next, we show $\mathsf{Hyb} \approx \mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}$. Let Sim be the (PPT) algorithm of Hyb for generating $\{\mathsf{pd}_i\}_{i\in S_{\mathsf{act}}}$, i.e., lines 9-15 of Hyb. Sim uses only the corrupted secret shares $\{\mathsf{sk}_i\}_{i\in S_{\mathsf{corr}}}$ (and other public values). Therefore, owing to the privacy of $\{0,1\}$-LSS (Def. 3.3), we obtain $(\{\mathsf{sk}_i\}_{i\in S_{\mathsf{corr}}} \leftarrow \mathsf{ShareGen}(\mathsf{sk})) \approx (\{\mathsf{sk}_i\}_{i\in S_{\mathsf{corr}}} \leftarrow \mathsf{ShareGen}(\mathbf{0}))$, which implies $\mathsf{Hyb} \approx \mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}$. □

Note. To achieve polynomially short decryption shares while satisfying correctness (Thm. 5.1), we require $q = \Omega((\frac{B_{\mathsf{flood}}}{p} + 1)N^{5.3})$ and $q = \lambda^{O(1)}$. These conditions are satisfied if $\frac{B_{\mathsf{flood}}}{p} = \lambda^{O(1)}$ and $N = \lambda^{O(1)}$. Moreover, this is consistent with the security condition in Thm. 5.2, i.e., $B_{\mathsf{flood}} = \Omega(2^\lambda B_{\mathsf{err}})$, when $p = B_{\mathsf{flood}}/F > B_{\mathsf{err}}$ for some $F = \lambda^{O(1)}$.

6 Our Threshold FHE based on Shamir's Secret Sharing

We propose our serverless ThFHE with short decryption shares based on Shamir-SS. We first describe our construction and threshold decryption procedure in Sect. 6.1. Next, we prove the correctness and security of our scheme in Sect. 6.2 and Sect. 6.3, respectively.

Algorithm 7: Our ThFHE based on Shamir-SS; $\mathsf{ThFHE} := (\mathsf{PP}, \mathsf{KeyGen}, \mathsf{ShareGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$.

$\underline{\mathsf{PP}(1^\lambda, 1^N)}$:
1 **return** $\mathsf{pp} := (n, m, p, q, Q = p \cdot q, \chi_{\mathsf{lwe}}, \sigma, B_{\mathsf{err}})$
▷ We let the following functions implicitly include pp as an argument

$\underline{\mathsf{KeyGen}()}$:
2 **return** $(\mathsf{evk}, \mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}()$ (Algo. 2)

$\underline{\mathsf{ShareGen}(\mathsf{sk}, T, N)}$:
3 $\{\mathbf{s}_i \in \mathbb{Z}_Q^n\}_{i\in[N]} \leftarrow \mathsf{Shamir}_{\mathbb{Z}_Q^n}(\mathsf{sk}, T, N)$ (Constr. 3.7)
4 **for** $(i,j) \in [N] \times [N]$ with $i \neq j$ **do** $\mathsf{seed}_{i,j} \xleftarrow{\$} \{0,1\}^\lambda$
5 **for** $i \in [N]$ **do** $\mathsf{seedset}_i := \{\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i}\}_{j\in[N]\setminus\{i\}}$
6 **return** $\{\mathsf{sk}_i := (\mathbf{s}_i, \mathsf{seedset}_i)\}_{i\in[N]}$

$\underline{\mathsf{Enc}(\mathsf{pk}, \mu \in \mathcal{M} := \{0,1\})}$:
7 **return** $\mathsf{ct} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, \mu)$ (Algo. 2)

$\underline{\mathsf{Eval}(\mathsf{evk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_l)}$:
8 **return** $\mathsf{ct}_{\mathsf{eval}} \leftarrow \mathsf{FHE.Eval}(\mathsf{evk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_l)$ (Algo. 2)

$\underline{\mathsf{PartDec}(\mathsf{ct} := (\mathbf{a}, b), \mathsf{sk}_i, \mathsf{sid}, S_{\mathsf{act}})}$: ▷ The arguments sid and S_{act} are required here
9 Parse $\mathsf{sk}_i := (\mathbf{s}_i, \mathsf{seedset}_i)$
10 $\Delta_i \leftarrow \mathsf{ZeroShare}(\mathsf{seedset}_i, \mathsf{ct} \,\|\, \mathsf{sid}, S_{\mathsf{act}})$
11 $\mathsf{pd}_i := \lfloor \frac{1}{p}(L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i \bmod Q) \rceil_\sigma + \Delta_i \bmod q$ ▷ $L_i^{S_{\mathsf{act}}}$: Constr. 3.7
12 **return** pd_i

$\underline{\mathsf{ZeroShare}(\mathsf{seedset}_i, x \in \{0,1\}^*, S_{\mathsf{act}})}$: ▷ Subroutine of PartDec
13 Parse $\mathsf{seedset}_i := \{\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i}\}_{j\in[N]\setminus\{i\}}$
14 **for** $j \in S_{\mathsf{act}} \setminus \{i\}$ **do**
15 $m_{i,j} \leftarrow \mathsf{PRF}(\mathsf{seed}_{i,j}, x) \in \mathbb{Z}_q$
16 $m_{j,i} \leftarrow \mathsf{PRF}(\mathsf{seed}_{j,i}, x) \in \mathbb{Z}_q$ ▷ $\mathsf{PRF}(\cdot,\cdot) : \{0,1\}^* \to \mathbb{Z}_q$ (Def. 3.15)
17 **return** $\Delta_i := \sum_{j\in S_{\mathsf{act}}\setminus\{i\}} (m_{i,j} - m_{j,i}) \bmod q$ ▷ $\sum_{i\in S_{\mathsf{act}}} \Delta_i = 0$ (cf., Fig. 4)

$\underline{\mathsf{FinDec}(\mathsf{ct} := (\mathbf{a}, b), \{\mathsf{pd}_i\}_{i\in S_{\mathsf{act}}})}$:
18 **assert** $\{|S_{\mathsf{act}}| \geq T\}$
19 **return** $\overline{\mu} := \lfloor (\lfloor \frac{b}{p} \rfloor - \sum_{i\in S_{\mathsf{act}}} \mathsf{pd}_i \bmod q) / \lfloor \frac{q}{2} \rfloor \rceil$

6.1 Construction

We present our construction of Shamir-SS-based ThFHE in Algo. 7. As in Algo. 5, we perform rounding during PartDec (line 11) to make the decryption share polynomially short. However, there are several technical changes compared to the previous $\{0,1\}$-LSS-based ThFHE (Algo. 5):

– *Synchronization*: Compared with $\{0,1\}$-LSS-based construction presented in Sect. 5.1, the Shamir-SS-based threshold decryption procedure requires one additional round of communication for "synchronization" to agree on

the active party set S_{act}. The parties should first declare whether they are active or not during the additional first round because our Shamir-SS-based construction requires knowing the active party set S_{act} of the decryption session.

- *Masking*: We apply the "zero-share" masking technique [KRT24] for security. The seed values $\mathsf{seed}_{i,j}$ are shared among parties, and they are used to generate "zero-share" mask values $\Delta_i \leftarrow \mathsf{ZeroShare}(\mathsf{seedset}_i, \mathsf{ct} \parallel \mathsf{sid}, S_{\mathsf{act}})$ (line 10), where $\mathsf{seedset}_i := \{\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i}\}_{j\in[N]\setminus\{i\}}$ and S_{act} is the set of active parties in the decryption session. The mask Δ_i is essentially pseudorandom except for the condition $\sum_{i\in S_{\mathsf{act}}} \Delta_i = 0$. We use Δ_i to make pd_i simulatable as a uniformly random value in the security proof.
- *Randomized Gaussian Rounding*: In the $\{0,1\}$-LSS-based construction presented in Sect. 5.1, the partial decryption is constructed with simple (deterministic) flooring $\lfloor\cdot\rfloor$ (line 7). In contrast, the Shamir-SS-based construction requires randomized Gaussian rounding $\lfloor\cdot\rceil_\sigma$ (line 11) defined in Def. 3.13. This is because, unlike the previous method in Sect. 5.1, we need to simulate the sum of the rounding errors in the partial decryption of honest parties. Further details are provided in the security proof.

Our *two*-round threshold decryption procedure is described as follows:

Input/output. Let a ciphertext ct and a session ID sid be inputs. As an output of threshold decryption, all parties recover the plaintext μ of ct.

1st round. Each party $P_i \in \{P_1, \ldots, P_N\}$ broadcasts 1 if P_i is "active" and 0 (or does nothing) otherwise. Then, at the end of the 1st round, the parties obtain an index set $S_{\mathsf{act}} \subseteq [N]$ of parties who are active.

2nd round. Each party P_i for $i \in S_{\mathsf{act}}$ performs $\mathsf{PartDec}$ on $\mathsf{ct} := (\mathbf{a}, b)$ and generates the polynomially short partial decryption share $\mathsf{pd}_i \in \mathbb{Z}_q$. Then, P_i broadcasts pd_i to all of the parties.

End of the 2nd round. Given $\{\mathsf{pd}_i\}_{i\in S_{\mathsf{act}}}$, all of the parties perform FinDec and obtain the plaintext μ under the condition of correctness (Thm. 6.1).

Note. Previous schemes, including [AJL+12; BGG+18; BS24; PS25; OT26a; ZZLP25], as well as our scheme, are essentially constructed in the *security-with-abort* model. For example, in [AJL+12], if a (malicious or *fail-corrupt*) party goes offline and does not send the decryption share, then (honest) parties fail to decrypt. Similarly, in our construction, if a (malicious or fail-corrupt) party goes offline between the 1st and the 2nd round, then (honest) parties fail to decrypt. These *security-with-abort* protocols can be improved to protocols with *fairness* or *guaranteed output delivery* by using generic techniques of, e.g., [AJL+12, Sect. G] or [GLS15].

6.2 Correctness

We prove the correctness (Def. 4.3) of Algo. 7:

Theorem 6.1. *Let $\epsilon = \mathrm{negl}(\lambda)$, $\sigma \geq \sqrt{2}\eta_\epsilon(\mathbb{Z})$, $p > B_{\mathsf{err}}$, $q = \sigma\sqrt{N} \cdot \Omega(\sqrt{\lambda})$, $Q := p \cdot q$, and assume that N and Q are chosen so that $i - j$ is invertible over $\mathbb{Z}_Q$ for all distinct $i, j \in [N]$. Then,* ThFHE *(Algo. 7) is correct (Def. 4.3) if the underlying* FHE *(Algo. 2) is correct.*

Proof. Our goal is to analyze the upper bound of the noise revealed after FinDec in line 19, i.e., $\lfloor \frac{b}{p} \rfloor - \sum_{i \in S_{\mathsf{act}}} \mathsf{pd}_i$. We first rewrite pd_i as follows:

$$\mathsf{pd}_i = \tfrac{1}{p} L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i + e_i + \Delta_i \bmod q, \text{ where } e_i \leftarrow \mathcal{D}_{\mathbb{Z} - \{\frac{1}{p} L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i\}, \sigma}.$$

Hence, by the correctness of Shamir-SS and construction of ZeroShare, we have

$$\begin{aligned} \textstyle\sum_{i \in S_{\mathsf{act}}} \mathsf{pd}_i \bmod q &= \tfrac{1}{p} (\underbrace{\textstyle\sum_{i \in S_{\mathsf{act}}} L_i^{S_{\mathsf{act}}} \mathbf{a}^\intercal \mathbf{s}_i}_{\in \mathbf{a}^\intercal \mathbf{s} + Q \cdot \mathbb{Z}}) + \textstyle\sum_{i \in S_{\mathsf{act}}} e_i + \underbrace{\textstyle\sum_{i \in S_{\mathsf{act}}} \Delta_i}_{=0} \bmod q, \\ &= \tfrac{1}{p} \mathbf{a}^\intercal \mathbf{s} + \textstyle\sum_{i \in S_{\mathsf{act}}} e_i \bmod q. \end{aligned} \tag{21}$$

By the hypothesis $\epsilon = \mathrm{negl}(\lambda)$ and $\sigma \geq \sqrt{2}\eta_\epsilon(\mathbb{Z})$, we can use Lem. 3.9 to obtain

$$\textstyle\sum_{i \in S_{\mathsf{act}}} e_i \approx_{\mathsf{s}} D_{\mathbb{Z} - \sum_{i \in S_{\mathsf{act}}} \{\frac{1}{p} L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i\}, \sqrt{|S_{\mathsf{act}}|}\sigma} = \mathcal{D}_{\mathbb{Z} - \{\frac{1}{p} \mathbf{a}^\intercal \mathbf{s}\}, \sqrt{|S_{\mathsf{act}}|}\sigma}. \tag{22}$$

Next, we analyze $\lfloor \frac{b}{p} \rfloor$ on line 19. The input ciphertext is expressed as follows:

$$\mathsf{ct} := (\mathbf{a} \in \mathbb{Z}_Q^n, b := \mathbf{a}^\intercal \mathbf{s} + e_{\mathsf{ct}} + \lfloor \tfrac{Q}{2} \rfloor \cdot \mu \bmod Q),$$

where e_{ct} satisfies $|e_{\mathsf{ct}}| < B_{\mathsf{err}}$ (as mentioned in Algo. 2). For this b, we obtain

$$\begin{aligned} \lfloor \tfrac{b}{p} \rfloor \bmod q &= \tfrac{1}{p} \mathbf{a}^\intercal \mathbf{s} + \tfrac{1}{p} e_{\mathsf{ct}} + \tfrac{1}{p} \lfloor \tfrac{Q}{2} \rfloor \cdot \mu - \{\tfrac{b}{p}\} \bmod q \\ &= \tfrac{1}{p} \mathbf{a}^\intercal \mathbf{s} + e' + \lfloor \tfrac{q}{2} \rfloor \cdot \mu \bmod q, \end{aligned} \tag{23}$$

where we define $e' := \frac{1}{p} e_{\mathsf{ct}} + (\{\frac{q}{2}\} - \frac{1}{p}\{\frac{Q}{2}\}) \cdot \mu - \{\frac{b}{p}\}$, and we used the fact that $\frac{1}{p}\lfloor \frac{Q}{2} \rfloor = \lfloor \frac{q}{2} \rfloor + \{\frac{q}{2}\} - \frac{1}{p}\{\frac{Q}{2}\}$. Finally, by (21), (22) and (23), we obtain

$$\begin{aligned} \lfloor \tfrac{b}{p} \rfloor - \textstyle\sum_{i \in S_{\mathsf{act}}} \mathsf{pd}_i \bmod q &= \lfloor \tfrac{q}{2} \rfloor \cdot \mu + e' - \textstyle\sum_{i \in S_{\mathsf{act}}} e_i \bmod q \\ &\approx_{\mathsf{s}} \lfloor \tfrac{q}{2} \rfloor \cdot \mu + e' - \mathcal{D}_{\mathbb{Z} - \{\frac{1}{p} \mathbf{a}^\intercal \mathbf{s}\}, \sqrt{|S_{\mathsf{act}}|}\sigma} \\ &= \lfloor \tfrac{q}{2} \rfloor \cdot \mu - \mathcal{D}_{\mathbb{Z} - \{\frac{1}{p} \mathbf{a}^\intercal \mathbf{s}\} - e', \sqrt{|S_{\mathsf{act}}|}\sigma, -e'} \\ &= \lfloor \tfrac{q}{2} \rfloor \cdot \mu - \mathcal{D}_{\mathbb{Z}, \sqrt{|S_{\mathsf{act}}|}\sigma, -e'} \end{aligned}$$

Note that $\mathbb{Z} - \{\frac{1}{p}\mathbf{a}^\intercal \mathbf{s}\} - e' = \mathbb{Z} - \{\frac{1}{p}\mathbf{a}^\intercal \mathbf{s} + e'\} = \mathbb{Z}$ since $\frac{1}{p}\mathbf{a}^\intercal \mathbf{s} + e' = \lfloor \frac{b}{p} \rfloor - \lfloor \frac{q}{2} \rfloor \cdot \mu \in \mathbb{Z}$. Since $|e_{\mathsf{ct}}| = O(B_{\mathsf{err}})$ and $p > B_{\mathsf{err}}$ by hypothesis, we have $|e'| = O(1)$. Thus, by Lem. 3.10, $\mathcal{D}_{\mathbb{Z}, \sqrt{|S_{\mathsf{act}}|}\sigma, e'}$ is $\sqrt{|S_{\mathsf{act}}|}\sigma \cdot \Omega(\sqrt{\lambda})$-bounded. Note that we have $|S_{\mathsf{act}}| \leq N$. Therefore, since $q = \sigma\sqrt{N} \cdot \Omega(\sqrt{\lambda})$ by hypothesis,

$$\overline{\mu} := \lfloor (\lfloor \tfrac{b}{p} \rfloor - \textstyle\sum_{i \in S_{\mathsf{act}}} \mathsf{pd}_i) / \lfloor \tfrac{q}{2} \rfloor \rceil = \mu$$

holds with overwhelming probability. □

6.3 Security

We prove the security (Def. 4.4) of Shamir-SS-based ThFHE (Algo. 7):

Theorem 6.2. *Assume that $\epsilon, p, q, B_{\mathsf{err}}, \sigma$ are selected as in Thm. 6.1. In addition, assume $p = \Omega(2^\lambda \cdot B_{\mathsf{err}})$. Then,* ThFHE *(Algo. 7) is simulation-secure (Def. 4.4) under the pseudorandomness assumption for* PRF*. Formally, for any PPT algorithm $\mathcal{A}$, there exist PPT algorithms* Sim *and $\mathcal{B}$ s.t.*

$$\mathrm{Adv}^{\mathsf{ThFHE}}_{\mathcal{A}}(\lambda) \leq \mathrm{Adv}^{\mathsf{PRF}}_{\mathcal{B}}(\lambda) + \mathrm{negl}(\lambda),$$

where $\mathrm{Adv}^{\mathsf{PRF}}_{\mathcal{B}}(\lambda)$ is defined as in Def. 3.15.

Proof. The full formal proof can be found in Appendix B. Here, we provide a proof overview. We define the hybrid experiments $\mathsf{Hyb}_1, \ldots, \mathsf{Hyb}_4$ and prove that

$$\mathsf{Expt}_{\mathcal{A},\mathsf{Real}} = \mathsf{Hyb}_1 \approx_{\mathsf{c}} \mathsf{Hyb}_2 \approx \mathsf{Hyb}_3 \approx_{\mathsf{s}} \mathsf{Hyb}_4 \approx \mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}$$

to obtain the claim. We also define $S^{\mathsf{hon}}_{\mathsf{act}} := S_{\mathsf{hon}} \cap S_{\mathsf{act}}$.

In Hyb_1 (Algo. 8), the challenger generates the partial decryption share pd_h of some fixed honest party $h \in S^{\mathsf{hon}}_{\mathsf{act}}$ without using the secret share $\mathbf{s}_h$ and the mask Δ_h of $h \in S^{\mathsf{hon}}_{\mathsf{act}}$ as follows:

$$\mathsf{pd}_h = \tfrac{1}{p}\left(b - e_{\mathsf{ct}} - \lfloor \tfrac{Q}{2} \rfloor \mu\right) - \sum_{i \in S_{\mathsf{act}} \setminus \{h\}} (\mathsf{pd}_i - e_i) + e_h \bmod q, \tag{24}$$

where e_i is the randomized rounding error in $\mathsf{pd}_i := \frac{1}{p}(L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i \bmod Q) + e_i + \Delta_i \bmod q$ for $i \in S_{\mathsf{act}}$. In $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$, by (21), we obtain

$$\begin{aligned}\textstyle\sum_{i \in S_{\mathsf{act}}} \mathsf{pd}_i \bmod q &= \tfrac{1}{p}\mathbf{a}^\intercal \mathbf{s} + \textstyle\sum_{i \in S_{\mathsf{act}}} e_i \bmod q. \\ &= \tfrac{1}{p}(b - e_{\mathsf{ct}} - \lfloor \tfrac{Q}{2} \rfloor \cdot \mu) + \textstyle\sum_{i \in S_{\mathsf{act}}} e_i \bmod q,\end{aligned}$$

and we obtain (24). Thus, pd_h in Hyb_1 is exactly equivalent to pd_h in $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$.

Next, in Hyb_2 (Algo. 9), mask values $m_{i,h} := \mathsf{PRF}(\mathsf{seed}_{i,h}, \mathsf{ct} \,\|\, \mathsf{sid}), m_{h,i} := \mathsf{PRF}(\mathsf{seed}_{h,i}, \mathsf{ct} \,\|\, \mathsf{sid})$ in Hyb_1, which are shared among honest parties h and $i \in S^{\mathsf{hon}}_{\mathsf{act}} \setminus \{h\}$, are replaced with uniformly random values. We obtain $\mathsf{Hyb}_1 \approx_{\mathsf{c}} \mathsf{Hyb}_2$ under the pseudorandomness assumption for PRF, since $m_{i,h}, m_{h,i}$ are only used to calculate Δ_i and are not used elsewhere.

Then, in Hyb_3 (Algo. 10), pd_i for $i \in S^{\mathsf{hon}}_{\mathsf{act}} \setminus \{h\}$, the partial decryption shares of all honest parties except for h are sampled uniformly at random. The uniformly random mask values $m_{i,h}, m_{h,i}$ for $i \in S^{\mathsf{hon}}_{\mathsf{act}} \setminus \{h\}$ are only used to calculate Δ_i in Hyb_2; thus, Δ_i is uniformly random from the adversary's view. Similarly, pd_i is also uniformly random from the adversary's view. Thus, we obtain $\mathsf{Hyb}_2 \approx \mathsf{Hyb}_3$.

Finally, in Hyb_4 (Algo. 11), pd_h is generated as follows:

$$\mathsf{pd}_h := \left\lfloor \tfrac{1}{p}\Big(b - \lfloor \tfrac{Q}{2} \rfloor \mu - \sum_{i \in S^{\mathsf{corr}}_{\mathsf{act}}} L_i^{S_{\mathsf{act}}} \mathbf{a}^\intercal \mathbf{s}_i\Big) \right\rceil_{\sqrt{|S^{\mathsf{hon}}_{\mathsf{act}}|}\sigma} - \sum_{i \in S^{\mathsf{hon}}_{\mathsf{act}} \setminus \{h\}} \mathsf{pd}_i - \sum_{i \in S^{\mathsf{corr}}_{\mathsf{act}}} \Delta_i \bmod q. \tag{25}$$

In Hyb_3, by definition of $\mathsf{pd}_i := \frac{1}{p}(L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i \bmod Q) + e_i + \Delta_i \bmod q$, we can rewrite (24) as follows:

$$\begin{aligned}\mathsf{pd}_h =& \frac{1}{p}\Big(b - \lfloor \frac{Q}{2} \rfloor \mu - \sum_{i \in S_{\mathsf{act}}^{\mathsf{corr}}} L_i^{S_{\mathsf{act}}} \mathbf{a}^\intercal \mathbf{s}_i\Big) - \frac{e_{\mathsf{ct}}}{p} + \sum_{i \in S_{\mathsf{act}}^{\mathsf{hon}}} e_i \\ & - \sum_{i \in S_{\mathsf{act}}^{\mathsf{hon}} \setminus \{h\}} \mathsf{pd}_i - \sum_{i \in S_{\mathsf{act}}^{\mathsf{corr}}} \Delta_i \bmod q \end{aligned} \tag{26}$$

By the hypothesis $\epsilon = \mathrm{negl}(\lambda)$ and $\sigma \geq \sqrt{2}\eta_\epsilon(\mathbb{Z})$, we can use Lem. 3.9 to obtain

$$\textstyle\sum_{i \in S_{\mathsf{act}}^{\mathsf{hon}}} e_i \approx_{\mathsf{s}} D_{\mathbb{Z} - \sum_{i \in S_{\mathsf{act}}^{\mathsf{hon}}} \{\frac{1}{p} L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i\}, \sqrt{|S_{\mathsf{act}}^{\mathsf{hon}}|}\sigma}.$$

Furthermore, according to the smudging lemma Cor. 3.12 and the hypothesis of $p = \Omega(2^\lambda B_{\mathsf{err}})$ and $|e_{\mathsf{ct}}| < B_{\mathsf{err}}$, we obtain

$$\textstyle -\frac{e_{\mathsf{ct}}}{p} + \sum_{i \in S_{\mathsf{act}}^{\mathsf{hon}}} e_i \approx_{\mathsf{s}} D_{\mathbb{Z} - (\frac{e_{\mathsf{ct}}}{p} + \sum_{i \in S_{\mathsf{act}}^{\mathsf{hon}}} \{\frac{1}{p} L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i\}), \sqrt{|S_{\mathsf{act}}^{\mathsf{hon}}|}\sigma}. \tag{27}$$

Additionally, by $\sum_{i \in S_{\mathsf{act}}^{\mathsf{hon}}} L_i^{S_{\mathsf{act}}} \mathbf{s}_i = \mathbf{s} - \sum_{i \in S_{\mathsf{act}}^{\mathsf{corr}}} L_i^{S_{\mathsf{act}}} \mathbf{s}_i$, we have

$$\frac{1}{p}\Big(b - \lfloor \frac{Q}{2} \rfloor \mu - \sum_{i \in S_{\mathsf{act}}^{\mathsf{corr}}} L_i^{S_{\mathsf{act}}} \mathbf{a}^\intercal \mathbf{s}_i\Big) = \frac{e_{\mathsf{ct}}}{p} + \sum_{i \in S_{\mathsf{act}}^{\mathsf{hon}}} \frac{1}{p} L_i^{S_{\mathsf{act}}} \mathbf{a}^\intercal \mathbf{s}_i. \tag{28}$$

Hence, by (26), (27), (28) (and Def. 3.13), we obtain

$$\mathsf{pd}_h \approx_{\mathsf{s}} \Big\lfloor \frac{1}{p}\Big(b - \lfloor \frac{Q}{2} \rfloor \mu - \sum_{i \in S_{\mathsf{act}}^{\mathsf{corr}}} L_i^{S_{\mathsf{act}}} \mathbf{a}^\intercal \mathbf{s}_i\Big) \Big\rceil_{\sqrt{|S_{\mathsf{act}}^{\mathsf{hon}}|}\sigma} - \sum_{i \in S_{\mathsf{act}}^{\mathsf{hon}} \setminus \{h\}} \mathsf{pd}_i - \sum_{i \in S_{\mathsf{act}}^{\mathsf{corr}}} \Delta_i \bmod q.$$

Thus, by (25) and the above, $\mathsf{Hyb}_3 \approx_{\mathsf{s}} \mathsf{Hyb}_4$ holds.

In Hyb_4, pd_h (defined in (25)) is generated only with the corrupted secret shares $\{\mathsf{sk}_i\}_{i \in S_{\mathsf{corr}}}$, and the other secret shares $\{\mathsf{sk}_i\}_{i \in S_{\mathsf{hon}}}$ are not used elsewhere. Hence, by the privacy of Shamir-SS, we have $\mathsf{Hyb}_4 \approx \mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}$. □

Acknowledgement. This work was supported by JST K Program Grant Number JPMJKP24U2, Japan.

References

[ABV+12] S. Agrawal, X. Boyen, V. Vaikuntanathan, P. Voulgaris, H. Wee. "Functional Encryption for Threshold Functions (or Fuzzy IBE) from Lattices". *PKC 2012*. 2012, pp. 280–297.

[ARS+15] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, M. Zohner. "Ciphers for MPC and FHE". *EUROCRYPT 2015*. 2015, pp. 430–454.

[AJL+12] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, D. Wichs. "Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE". *EUROCRYPT 2012*. 2012, pp. 483–501.

[BCK+23] Y. Bae, J. H. Cheon, J. Kim, J. H. Park, D. Stehlé. "HERMES: Efficient Ring Packing Using MLWE Ciphertexts and Application to Transciphering". *CRYPTO 2023*. 2023, pp. 37–69.

[BGG+18] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, A. Sahai. "Threshold Cryptosystems from Threshold Fully Homomorphic Encryption". *CRYPTO 2018*. 2018, pp. 565–596.

[BDE+18] J. Bootle, C. Delaplace, T. Espitau, P.-A. Fouque, M. Tibouchi. "LWE Without Modular Reduction and Improved Side-Channel Attacks Against BLISS". *ASIACRYPT 2018*. 2018, pp. 494–524.

[BJTW25] K. Boudgoust, C. Jeudy, E. Tairi, W. Wen. *Hardness of M-LWE with General Distributions and Applications to Leaky Variants*. Cryptology ePrint Archive, Paper 2025/1472. 2025.

[BS24] K. Boudgoust, P. Scholl. *Simple Threshold (Fully Homomorphic) Encryption From LWE With Polynomial Modulus*. ePrint 2023/016. 2024. Version 2024-07-16.

[BGV12] Z. Brakerski, C. Gentry, V. Vaikuntanathan. "(Leveled) Fully Homomorphic Encryption without Bootstrapping". *ITCS 2012*. ACM, 2012, pp. 309–325.

[BP25] L. T. Brandão, R. Peralta. *NIST First Call for Multi-Party Threshold Schemes*. NISTIR 8214C 2pd (Second Public Draft). 2025.

[CHK+18] J. H. Cheon, K. Han, A. Kim, M. Kim, Y. Song. "Bootstrapping for Approximate Homomorphic Encryption". *EUROCRYPT 2018*. 2018, pp. 360–384.

[DPSZ12] I. Damgård, V. Pastro, N. Smart, S. Zakarias. "Multiparty Computation from Somewhat Homomorphic Encryption". *CRYPTO 2012*. 2012, pp. 643–662.

[GGHR14] S. Garg, C. Gentry, S. Halevi, M. Raykova. "Two-Round Secure MPC from Indistinguishability Obfuscation". *TCC 2014*. 2014, pp. 74–94.

[Gen09] C. Gentry. "Fully Homomorphic Encryption Using Ideal Lattices". *STOC '09*. ACM, 2009, pp. 169–178.

[GSW13] C. Gentry, A. Sahai, B. Waters. "Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based". *CRYPTO 2013*. 2013, pp. 75–92.

[Gol20] O. Goldreich. "On (Valiant's) Polynomial-Size Monotone Formula for Majority". *Computational Complexity and Property Testing*. 2020, pp. 17–23.

[GLS15] S. Dov Gordon, F.-H. Liu, E. Shi. "Constant-Round MPC with Fairness and Guarantee of Output Delivery". *CRYPTO 2015*. 2015, pp. 63–82.

[HKL+22] J. Ha, S. Kim, B. Lee, J. Lee, M. Son. "Rubato: Noisy Ciphers for Approximate Homomorphic Encryption". *EUROCRYPT 2022*. 2022, pp. 581–610.

[KRT24] S. Katsumata, M. Reichle, K. Takemure. "Adaptively Secure 5 Round Threshold Signatures from MLWE/MSIS and DL with Rewinding". *CRYPTO 2024*. 2024, pp. 459–491.

[LP11] R. Lindner, C. Peikert. "Better Key Sizes (and Attacks) for LWE-Based Encryption". *CT-RSA 2011*. 2011, pp. 319–339.

[MP13] D. Micciancio, C. Peikert. "Hardness of SIS and LWE with Small Parameters". *CRYPTO 2013*. 2013, pp. 21–39.

[MS25] D. Micciancio, A. Suhl. "Simulation-Secure Threshold PKE from LWE with Polynomial Modulus". *IACR Communications in Cryptology* 1.4 (Jan. 13, 2025).

[MW16] P. Mukherjee, D. Wichs. "Two Round Multiparty Computation via Multi-key FHE". *EUROCRYPT 2016*. 2016, pp. 735–763.

[NLV11] M. Naehrig, K. Lauter, V. Vaikuntanathan. "Can homomorphic encryption be practical?" *CCSW '11*. ACM, 2011, pp. 113–124.

[OT26a] H. Okada, T. Takagi. "Low Communication Threshold FHE from Standard (Module-)LWE". *ASIACRYPT 2025*. 2026, pp. 165–198.

[OT26b] H. Okada, T. Takagi. "Threshold FHE with Short Decryption Shares without a Semi-trusted Server". *ACISP 2026 (to appear)*. 2026.

[PS25] A. Passelègue, D. Stehlé. "Low Communication Threshold Fully Homomorphic Encryption". *ASIACRYPT 2024*. 2025, pp. 297–329.

[PKM+24] R. del Pino, S. Katsumata, M. Maller, F. Mouhartem, T. Prest, M.-J. Saarinen. "Threshold Raccoon: Practical Threshold Signatures from Standard Lattice Assumptions". *EUROCRYPT 2024*. 2024, pp. 219–248.

[Val84] L. Valiant. "Short monotone formulae for the majority function". *Journal of Algorithms* 5.3 (1984), pp. 363–366.

[ZZLP25] G. Zyskind, D. Zarchy, M. Leibovich, C. Peikert. "High-Throughput Universally Composable Threshold FHE Decryption". ACM CCS '25. 2025, pp. 2339–2353.

A Changelog

This paper has been updated since its initial appearance. We briefly summarize the changes below.

A.1 Version 1

Preliminary version to appear at ACISP 2026 [OT26b].

A.2 Version 2 (this version): Major revision

The preliminary version [OT26b] presented two ThFHE schemes. The first scheme is based on $\{0,1\}$-LSS ([OT26b, Section 5]), and the second scheme is based on Shamir-SS ([OT26b, Section 6]).

We have found a flaw in the security proof of the Shamir-SS-based scheme. Specifically, the proof of [OT26b, Section 6] is valid only in the restricted setting where the number of active parties is minimal, i.e., $|S_{\mathsf{act}}| = T$, and the adversary corrupts (maximal) $T-1$ *active* parties, i.e., $|S^{\mathsf{corr}}_{\mathsf{act}}| = T-1$ for $S^{\mathsf{corr}}_{\mathsf{act}} := S_{\mathsf{corr}} \cap S_{\mathsf{act}}$. In this case, there is only one honest active party, i.e., $|S^{\mathsf{hon}}_{\mathsf{act}}| = 1$ for $S^{\mathsf{hon}}_{\mathsf{act}} := S_{\mathsf{hon}} \cap S_{\mathsf{act}}$, and thus we can simulate the decryption share of the honest party as in the proof of [OT26b, Section 6]. However, when the number of active parties exceeds the minimal threshold, i.e., $|S_{\mathsf{act}}| > T$, the technique of [OT26b, Section 6] fails to simulate multiple decryption shares because the Lagrange coefficients are "embedded" in the decryption shares (see Sect. 2.5 for more details).

To address this issue, we revise the construction in Sect. 6 of the present version. In particular, we incorporate a *masking technique* to ensure that multiple decryption shares can be simulated as pseudorandom values. In addition, we employ a randomized Gaussian rounding technique, which enables the simulation of the sum of rounding errors required in the proof. As a result, we obtain a correct security proof for the Shamir-SS-based scheme.

B Full proof of Thm. 6.2

In this section, we provide the full formal proof of Thm. 6.2. We define the hybrid experiments $\mathsf{Hyb}_1, \ldots, \mathsf{Hyb}_4$ and prove Thm. 6.2 via Lem. B.1–B.5.

We define a hybrid experiment Hyb_1 in Algo. 8. In Hyb_1, the challenger generates the partial decryption share pd_h of some honest party $h \in S^{\mathsf{hon}}_{\mathsf{act}}$ without using the secret share $\mathbf{s}_h$ and the mask Δ_h of $h \in S^{\mathsf{hon}}_{\mathsf{act}}$. Although all other partial decryption shares $\{\mathsf{pd}_i\}_{i \in S_{\mathsf{act}} \setminus \{h\}}$ are the same as those in $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$, we explicitly define the rounding error e_i of the randomized Gaussian rounding here; we denote $\mathsf{pd}_i := \lfloor \frac{1}{p}(L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i \bmod Q) \rceil_\sigma + \Delta_i \bmod q = \frac{1}{p}(L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i \bmod Q) + e_i + \Delta_i \bmod q$, where $e_i \leftarrow \mathcal{D}_{\mathbb{Z}-\{\frac{1}{p}L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i\},\sigma}$. We show that Hyb_1 is equivalent to $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$.

Lemma B.1 ($\mathsf{Expt}_{\mathcal{A},\mathsf{Real}} \Leftrightarrow \mathsf{Hyb}_1$)**.** *The difference between* $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$ *(in Algo. 4) and* Hyb_1 *(Algo. 8) is purely conceptual. Formally, for any PPT adversary* $\mathcal{A}$, $\mathrm{Adv}^{\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}\text{-}\mathsf{Hyb}_1}_{\mathcal{A}} = 0$.

Proof. Let us write the input ciphertext ct as follows:

$$\mathsf{ct} := (\mathbf{a} \in \mathbb{Z}_Q^n, b := \mathbf{a}^\intercal \mathbf{s} + e_{\mathsf{ct}} + \lfloor \tfrac{Q}{2} \rfloor \cdot \mu \bmod Q).$$

In $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$, by (21), we have

$$\textstyle\sum_{i \in S_{\mathsf{act}}} \mathsf{pd}_i \bmod q = \frac{1}{p}\mathbf{a}^\intercal \mathbf{s} + \sum_{i \in S_{\mathsf{act}}} e_i \bmod q,$$

where $e_i \leftarrow \mathcal{D}_{\mathbb{Z}-\{\frac{1}{p}L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i\},\sigma}$ for $i \in S_{\mathsf{act}}$. Then, we obtain

$$\begin{aligned} 0 &= \tfrac{1}{p}\left(b - \lfloor \tfrac{Q}{2} \rfloor \mu - e_{\mathsf{ct}}\right) - \textstyle\sum_{i \in S_{\mathsf{act}}} (\mathsf{pd}_i - e_i) \bmod q, \\ \Leftrightarrow \mathsf{pd}_h &= \tfrac{1}{p}\left(b - \lfloor \tfrac{Q}{2} \rfloor \mu - e_{\mathsf{ct}}\right) - \sum_{i \in S_{\mathsf{act}} \setminus \{h\}} (\mathsf{pd}_i - e_i) + e_h \bmod q. \end{aligned}$$

Algorithm 8: Hyb_1 for the proof of Thm. 6.2

$\underline{\mathsf{Hyb}_1}$:

▷ Expt, $\mathcal{O}_{\mathsf{Enc}}$ and $\mathcal{O}_{\mathsf{Eval}}$ are identical to those in $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$ (Algo. 4)

$\underline{\mathcal{O}_{\mathsf{Dec}}(\mathsf{idx}, S_{\mathsf{act}})}$:

1 **assert** $\{|S_{\mathsf{act}}| \geq T$ and $\mathsf{idx} \leq |\mathsf{List}|\}$, $\mathsf{sid} \leftarrow \mathsf{sid} + 1$

2 $(\mu, \mathsf{ct}) \leftarrow \mathbf{pop}(\mathsf{List}, \mathsf{idx})$, and denote $\mathsf{ct} := (\mathbf{a}, b := \mathbf{a}^\intercal \mathbf{s} + e_{\mathsf{ct}} + \lfloor \frac{Q}{2} \rfloor \cdot \mu)$.

3 $S^{\mathsf{corr}}_{\mathsf{act}} := S_{\mathsf{corr}} \cap S_{\mathsf{act}}$, $S^{\mathsf{hon}}_{\mathsf{act}} := S_{\mathsf{hon}} \cap S_{\mathsf{act}}$

4 Fix some $h \in S^{\mathsf{hon}}_{\mathsf{act}}$

5 **for** $i \in S_{\mathsf{act}} \setminus \{h\}$ **do** ▷ Same as in $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$

6 $\quad \mathsf{pd}_i := \frac{1}{p}(L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i \bmod Q) + e_i + \Delta_i \bmod q \leftarrow \mathsf{PartDec}(\mathsf{ct}, \mathsf{sk}_i, \mathsf{sid}, S_{\mathsf{act}})$, where $e_i \leftarrow \mathcal{D}_{\mathbb{Z}-\{\frac{1}{p} L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i\}, \sigma}$

7 $\mathsf{pd}_h = \frac{1}{p}\left(b - e_{\mathsf{ct}} - \lfloor \frac{Q}{2} \rfloor \mu\right) - \sum_{i \in S_{\mathsf{act}} \setminus \{h\}} (\mathsf{pd}_i - e_i) + e_h \bmod q$, where $e_h \leftarrow \mathcal{D}_{\mathbb{Z}-\{\frac{1}{p} L_h^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_h\}, \sigma}$

8 **return** $(\mathsf{ct}, \{\mathsf{pd}_i\}_{i \in S_{\mathsf{act}}})$

Algorithm 9: Hyb_2 for the proof of Thm. 6.2.

$\underline{\mathsf{Hyb}_2}$:

▷ Expt, $\mathcal{O}_{\mathsf{Enc}}$ and $\mathcal{O}_{\mathsf{Eval}}$ are identical to those in $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$ of Algo. 4

$\underline{\mathcal{O}_{\mathsf{Dec}}(\mathsf{idx}, S_{\mathsf{act}})}$:

▷ Perform lines 1-3 of Hyb_1 (Algo. 8)

1 Fix some $h \in S^{\mathsf{hon}}_{\mathsf{act}}$

2 **for** $i \in S^{\mathsf{corr}}_{\mathsf{act}}$ **do** ▷ Same as in $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$

3 $\quad \mathsf{pd}_i := \frac{1}{p}(L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i \bmod Q) + e_i + \Delta_i \bmod q \leftarrow \mathsf{PartDec}(\mathsf{ct}, \mathsf{sk}_i, \mathsf{sid}, S_{\mathsf{act}})$

4 **for** $i \in S^{\mathsf{hon}}_{\mathsf{act}} \setminus \{h\}$ **do**

5 $\quad m_{i,h}, m_{h,i} \xleftarrow{\$} \mathbb{Z}_q$ ▷ Uniformly random

6 $\quad$ **for** $j \in S_{\mathsf{act}} \setminus \{i, h\}$ **do**

$\quad\quad m_{i,j} := \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{ct} \,\|\, \mathsf{sid}), m_{j,i} := \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{ct} \,\|\, \mathsf{sid})$

7 $\quad \Delta_i := \sum_{j \in S_{\mathsf{act}} \setminus \{i\}} (m_{i,j} - m_{j,i}) \mod q$

8 $\quad e_i \leftarrow \mathcal{D}_{\mathbb{Z}-\{\frac{1}{p} L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i\}, \sigma}$

9 $\quad \mathsf{pd}_i := \frac{1}{p}(L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i \bmod Q) + e_i + \Delta_i \bmod q$

10 $e_h \leftarrow \mathcal{D}_{\mathbb{Z}-\{\frac{1}{p} L_h^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_h\}, \sigma}$

11 $\mathsf{pd}_h = \frac{1}{p}\left(b - e_{\mathsf{ct}} - \lfloor \frac{Q}{2} \rfloor \mu\right) - \sum_{i \in S_{\mathsf{act}} \setminus \{h\}} (\mathsf{pd}_i - e_i) + e_h \bmod q$

12 **return** $(\mathsf{ct}, \{\mathsf{pd}_i\}_{i \in S_{\mathsf{act}}})$

Since the right-hand side of the above is equal to pd_h in line 7 of Hyb_1, we obtain the claim. □

Next, in Hyb_2 (Algo. 9), mask values $m_{i,h}$ and $m_{h,i}$ shared among honest parties h and $i \in S^{\mathsf{hon}}_{\mathsf{act}} \setminus \{h\}$, are replaced with uniformly random values (in line 5).

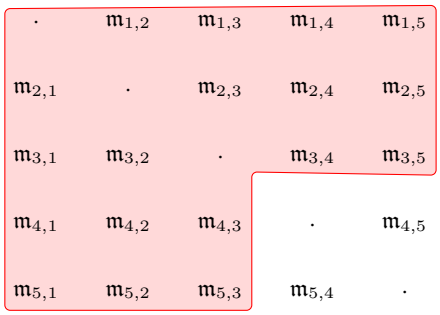


Fig. 4. Graphical explanation of ZeroShare: Here, $S_{\mathsf{act}} = \{1,\dots,5\}$, $\mathfrak{m}_{i,j} = \mathsf{PRF}(\mathsf{seed}_{i,j},\cdot)$. An adversary corrupting the party set $S_{\mathsf{corr}} = \{1,\dots,3\}$ learns the masks $\{\mathfrak{m}_{i,j}\}_{\min(i,j)\leq 3}$ (shaded). However, $\mathfrak{m}_{4,5}$ and $\mathfrak{m}_{5,4}$, which are known only to honest parties $S_{\mathsf{hon}} = \{4,5\}$, remain pseudorandom to the adversary (if PRF is a pseudorandom function). Thus, $\Delta_i := \sum_{j\in S_{\mathsf{act}}\setminus\{i\}}(m_{i,j} - m_{j,i})$ for $i \in S_{\mathsf{hon}} = \{4,5\}$ is essentially pseudorandom to the adversary, except for the fact that $\sum_{i\in S_{\mathsf{act}}} \Delta_i = 0$.

By construction of ZeroShare (see Fig. 4 for graphical explanation), mask values $m_{i,h}$ and $m_{h,i}$ are known only to honest parties. Hence, we obtain $\mathsf{Hyb}_1 \approx_{\mathsf{c}} \mathsf{Hyb}_2$ under the pseudorandomness assumption for PRF;

Lemma B.2 ($\mathsf{Hyb}_1 \approx_{\mathsf{c}} \mathsf{Hyb}_2$)**.** *The views of the adversary in* Hyb_1 *(Algo. 8) and* Hyb_2 *(Algo. 9) are computationally indistinguishable under the pseudorandomness assumption for* PRF*. Formally, for any PPT adversary* $\mathcal{A}$*, there exists a PPT algorithm* $\mathcal{B}$*, s.t.* $\mathrm{Adv}^{\mathsf{Hyb}_1\text{-}\mathsf{Hyb}_2}_{\mathcal{A}}(\lambda) \leq \mathrm{Adv}^{\mathsf{PRF}}_{\mathcal{B}}(\lambda)$*, where* $\mathrm{Adv}^{\mathsf{PRF}}_{\mathcal{B}}(\lambda)$ *is defined as in Def. 3.15.*

Proof. In Hyb_1 (Algo. 8), the zero-share mask $\Delta_h := \sum_{j\in S_{\mathsf{act}}\setminus\{h\}}(m_{h,j} - m_{j,h})$ of the honest party $h \in S^{\mathsf{hon}}_{\mathsf{act}}$ (fixed in line 4) is not used anywhere. Thus, for any i, the mask values $m_{i,h}, m_{h,i}$ are calculated by the party i. Furthermore, for $i \in S^{\mathsf{hon}}_{\mathsf{act}} \setminus \{h\}$, $m_{i,h}, m_{h,i}$ are known only to honest parties. Hence, the only difference between Hyb_1 and Hyb_2 is whether $m_{i,h} := \mathsf{PRF}(\mathsf{seed}_{i,h}, \mathsf{ct} \,\|\, \mathsf{sid}), m_{h,i} := \mathsf{PRF}(\mathsf{seed}_{h,i}, \mathsf{ct} \,\|\, \mathsf{sid})$ or $m_{i,h}, m_{h,i} \xleftarrow{\$} \mathbb{Z}_q$. These are computationally indistinguishable under the pseudorandomness assumption for PRF. □

In Hyb_3 (Algo. 10), the partial decryption shares of all honest parties except for h are sampled uniformly at random in line 4.

Lemma B.3 ($\mathsf{Hyb}_2 \approx \mathsf{Hyb}_3$)**.** *The views of the adversary in* Hyb_2 *(Algo. 9) and* Hyb_3 *(Algo. 10) are identically distributed.*

Proof. In Hyb_2, since Δ_h is not used anywhere, the mask values $m_{i,h}, m_{h,i} \xleftarrow{\$} \mathbb{Z}_q$ are only used to calculate Δ_i. Hence, for $i \in S^{\mathsf{hon}}_{\mathsf{act}} \setminus \{h\}$, $\Delta_i := \sum_{j\in S_{\mathsf{act}}\setminus\{i\}}(m_{i,j} - m_{j,i})$ is uniformly random from the adversary's view. Consequently, pd_i is also uniformly random from the adversary's view since Δ_i is only used to calculate the partial decryption share pd_i. Thus, we obtain the claim. □

Algorithm 10: Hyb_3 for the proof of Thm. 6.2.

$\underline{\mathsf{Hyb}_3}$:

▷ Expt, $\mathcal{O}_{\mathsf{Enc}}$ and $\mathcal{O}_{\mathsf{Eval}}$ are identical to those in $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$ of Algo. 4

$\underline{\mathcal{O}_{\mathsf{Dec}}(\mathsf{idx}, S_{\mathsf{act}})}$:

▷ Perform lines 1-3 of Hyb_1 (Algo. 8)

1 Fix some $h \in S_{\mathsf{act}}^{\mathsf{hon}}$

2 **for** $i \in S_{\mathsf{act}}^{\mathsf{corr}}$ **do** ▷ Same as in $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$

3 $\quad \mathsf{pd}_i := \frac{1}{p}(L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i \bmod Q) + e_i + \Delta_i \bmod q \leftarrow \mathsf{PartDec}(\mathsf{ct}, \mathsf{sk}_i, \mathsf{sid}, S_{\mathsf{act}})$

4 **for** $i \in S_{\mathsf{act}}^{\mathsf{hon}} \setminus \{h\}$ **do** $e_i \leftarrow \mathcal{D}_{\mathbb{Z}-\{\frac{1}{p}L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i\},\sigma}$, $\mathsf{pd}_i \xleftarrow{\$} \mathbb{Z}_q$ ▷ Uniformly random

5 $e_h \leftarrow \mathcal{D}_{\mathbb{Z}-\{\frac{1}{p}L_h^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_h\},\sigma}$

6 $\mathsf{pd}_h = \frac{1}{p}\left(b - e_{\mathsf{ct}} - \lfloor \frac{Q}{2} \rfloor \mu\right) - \sum_{i \in S_{\mathsf{act}} \setminus \{h\}} (\mathsf{pd}_i - e_i) + e_h \bmod q$

7 **return** $(\mathsf{ct}, \{\mathsf{pd}_i\}_{i \in S_{\mathsf{act}}})$

Algorithm 11: Hyb_4 for the proof of Thm. 6.2.

$\underline{\mathsf{Hyb}_4}$:

▷ Expt, $\mathcal{O}_{\mathsf{Enc}}$ and $\mathcal{O}_{\mathsf{Eval}}$ are identical to those in $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$ of Algo. 4

$\underline{\mathcal{O}_{\mathsf{Dec}}(\mathsf{idx}, S_{\mathsf{act}})}$:

▷ Perform lines 1-3 of Hyb_1 (Algo. 8)

1 Fix some $h \in S_{\mathsf{act}}^{\mathsf{hon}}$

2 **for** $i \in S_{\mathsf{act}}^{\mathsf{corr}}$ **do** ▷ Same as in $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}$

3 $\quad \mathsf{pd}_i := \frac{1}{p}(L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i \bmod Q) + e_i + \Delta_i \bmod q \leftarrow \mathsf{PartDec}(\mathsf{ct}, \mathsf{sk}_i, \mathsf{sid}, S_{\mathsf{act}})$

4 **for** $i \in S_{\mathsf{act}}^{\mathsf{hon}} \setminus \{h\}$ **do** $\mathsf{pd}_i \xleftarrow{\$} \mathbb{Z}_q$

5 $\mathsf{pd}_h := \left\lfloor \frac{1}{p}\left(b - \lfloor \frac{Q}{2} \rfloor \mu - \sum_{i \in S_{\mathsf{act}}^{\mathsf{corr}}} L_i^{S_{\mathsf{act}}} \mathbf{a}^\intercal \mathbf{s}_i\right) \right\rceil_{\sqrt{|S_{\mathsf{act}}^{\mathsf{hon}}|}\sigma} - \sum_{i \in S_{\mathsf{act}}^{\mathsf{hon}} \setminus \{h\}} \mathsf{pd}_i - \sum_{i \in S_{\mathsf{act}}^{\mathsf{corr}}} \Delta_i \bmod q$

6 **return** $(\mathsf{ct}, \{\mathsf{pd}_i\}_{i \in S_{\mathsf{act}}})$

Then, in Hyb_4 (Algo. 11), the decryption share pd_h (line 5) is constructed only with secret shares of corrupted parties $\{\mathsf{sk}_i\}_{i \in S_{\mathsf{act}}^{\mathsf{corr}}}$ and publicly samplable values.

Lemma B.4 ($\mathsf{Hyb}_3 \approx_{\mathsf{s}} \mathsf{Hyb}_4$). *Assume $\epsilon = \mathrm{negl}(\lambda)$ and $\sigma \geq \sqrt{2}\eta_\epsilon(\mathbb{Z})$. Then, the views of the adversary in Hyb_3 (Algo. 10) and Hyb_4 (Algo. 11) are statistically indistinguishable.*

Proof. In Hyb_3 (Algo. 10), we have

$$\begin{aligned}\mathsf{pd}_h :=& \frac{1}{p}\left(b - \lfloor \frac{Q}{2} \rfloor \mu - e_{\mathsf{ct}}\right) - \sum_{i \in S_{\mathsf{act}} \setminus \{h\}} (\mathsf{pd}_i - e_i) + e_h \bmod q \\ :=& \frac{1}{p}\left(b - \lfloor \frac{Q}{2} \rfloor \mu - \sum_{i \in S^{\mathsf{corr}}_{\mathsf{act}}} L_i^{S_{\mathsf{act}}} \mathbf{a}^\intercal \mathbf{s}_i\right) - \frac{e_{\mathsf{ct}}}{p} + \sum_{i \in S^{\mathsf{hon}}_{\mathsf{act}}} e_i \\ & - \sum_{i \in S^{\mathsf{hon}}_{\mathsf{act}} \setminus \{h\}} \mathsf{pd}_i - \sum_{i \in S^{\mathsf{corr}}_{\mathsf{act}}} \Delta_i \bmod q \end{aligned} \tag{29}$$

By the hypothesis $\epsilon = \mathrm{negl}(\lambda)$ and $\sigma \geq \sqrt{2}\eta_\epsilon(\mathbb{Z})$, we can use Lem. 3.9 to obtain

$$\textstyle\sum_{i \in S^{\mathsf{hon}}_{\mathsf{act}}} e_i \approx_{\mathsf{s}} D_{\mathbb{Z} - \sum_{i \in S^{\mathsf{hon}}_{\mathsf{act}}} \{\frac{1}{p} L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i\}, \sqrt{|S^{\mathsf{hon}}_{\mathsf{act}}|}\sigma}.$$

Furthermore, by the smudging lemma Cor. 3.12 and the hypothesis of $p = \Omega(2^\lambda B_{\mathsf{err}})$ and $|e_{\mathsf{ct}}| < B_{\mathsf{err}}$, we obtain

$$\begin{aligned} -\frac{e_{\mathsf{ct}}}{p} + \textstyle\sum_{i \in S^{\mathsf{hon}}_{\mathsf{act}}} e_i &\approx_{\mathsf{s}} D_{\mathbb{Z} - (\frac{e_{\mathsf{ct}}}{p} + \sum_{i \in S^{\mathsf{hon}}_{\mathsf{act}}} \{\frac{1}{p} L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i\}), \sqrt{|S^{\mathsf{hon}}_{\mathsf{act}}|}\sigma, -\frac{e_{\mathsf{ct}}}{p}} \\ &\approx_{\mathsf{s}} D_{\mathbb{Z} - (\frac{e_{\mathsf{ct}}}{p} + \sum_{i \in S^{\mathsf{hon}}_{\mathsf{act}}} \{\frac{1}{p} L_i^{S_{\mathsf{act}}} \cdot \mathbf{a}^\intercal \mathbf{s}_i\}), \sqrt{|S^{\mathsf{hon}}_{\mathsf{act}}|}\sigma}. \end{aligned} \tag{30}$$

Also note that, by $\sum_{i \in S^{\mathsf{hon}}_{\mathsf{act}}} L_i^{S_{\mathsf{act}}} \mathbf{s}_i = \mathbf{s} - \sum_{i \in S^{\mathsf{corr}}_{\mathsf{act}}} L_i^{S_{\mathsf{act}}} \mathbf{s}_i$, we have

$$\frac{1}{p}\left(b - \lfloor \frac{Q}{2} \rfloor \mu - \sum_{i \in S^{\mathsf{corr}}_{\mathsf{act}}} L_i^{S_{\mathsf{act}}} \mathbf{a}^\intercal \mathbf{s}_i\right) = \frac{e_{\mathsf{ct}}}{p} + \sum_{i \in S^{\mathsf{hon}}_{\mathsf{act}}} \frac{1}{p} L_i^{S_{\mathsf{act}}} \mathbf{a}^\intercal \mathbf{s}_i. \tag{31}$$

Hence, by (29), (30), (31) (and Def. 3.13), in Hyb_3, we obtain

$$\mathsf{pd}_h \approx_{\mathsf{s}} \left\lfloor \frac{1}{p}\left(b - \lfloor \frac{Q}{2} \rfloor \mu - \sum_{i \in S^{\mathsf{corr}}_{\mathsf{act}}} L_i^{S_{\mathsf{act}}} \mathbf{a}^\intercal \mathbf{s}_i\right) \right\rceil_{\sqrt{|S^{\mathsf{hon}}_{\mathsf{act}}|}\sigma} - \sum_{i \in S^{\mathsf{hon}}_{\mathsf{act}} \setminus \{h\}} \mathsf{pd}_i - \sum_{i \in S^{\mathsf{corr}}_{\mathsf{act}}} \Delta_i \bmod q.$$

The right-hand side of the above is equal to pd_h in Hyb_4 (line 5). □

Finally, we show $\mathsf{Hyb}_4 \approx \mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}$ to complete the proof of Thm. 6.2.

Lemma B.5. *The views of the adversary in* Hyb_4 *(Algo. 11) and* $\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}$ *in Algo. 4 are identically distributed.*

Proof. Let Sim be the (PPT) procedure of Hyb_4 for outputting $\{\mathsf{pd}_i\}_{i \in S_{\mathsf{act}}}$, i.e., lines 1-5 of Hyb_4. Sim only uses the corrupted secret shares $\{\mathsf{sk}_i\}_{i \in S_{\mathsf{corr}}}$ (and other public values). Hence, by the privacy of Shamir-SS (Constr. 3.7), $\{\mathsf{sk}_i\}_{i \in S_{\mathsf{corr}}} \leftarrow \mathsf{ShareGen}(\mathsf{sk})$ and $\{\mathsf{sk}_i\}_{i \in S_{\mathsf{corr}}} \leftarrow \mathsf{ShareGen}(\mathbf{0})$ are identically distributed. □