

A Critical Look into Threshold Homomorphic Encryption for Private Average Aggregation

Miguel Morona-Mínguez, Alberto Pedrouzo-Ulloa and Fernando Pérez-González
atlanTTic, Universidade de Vigo
{mmorona, apedrouzo, fperez}@gts.uvigo.es

Abstract—Threshold Homomorphic Encryption (Threshold HE) is a good fit for implementing private federated average aggregation, a key operation in Federated Learning (FL). Despite its potential, recent studies have shown that threshold schemes available in mainstream HE libraries can introduce unexpected security vulnerabilities if an adversary has access to a restricted decryption oracle. This oracle reflects the FL clients’ capacity to collaboratively decrypt the aggregated result without knowing the secret key. This work surveys the use of threshold RLWE-based HE for federated average aggregation and examines the performance impact of using smudging noise with a large variance as a countermeasure. We provide a detailed comparison of threshold variants of BFV and CKKS, finding that CKKS-based aggregations perform comparably to BFV-based solutions.

Index Terms—Machine Learning, Federated Learning, Private Aggregation, Homomorphic Encryption, Multiple Keys.

I. INTRODUCTION

Federated Learning (FL) was introduced in [1], [2] as a Machine Learning (ML) setting where, under the coordination of a central server, multiple clients collaborate with the aim of solving a learning task. One distinguishing feature of FL relies on the fact that the data used for training is distributed among the different clients, in such a way that datasets remain separate and stored locally in each of the clients’ premises.

The learning process (see Fig. 1) is as follows: Starting from an initial common ML model $\vec{w}^*$, an FL protocol typically consists in a series of rounds in which, (1) the clients locally update the common model by partially training with only their own individual datasets, obtaining a local update $\vec{w}^i$, and (2) the central server or aggregator, combines all received local updates from the clients to obtain a new common ML model, which is done by computing an aggregation function $\vec{w}^* = f_{\Sigma}(\vec{w}^1, \dots, \vec{w}^L)$. This process can be repeated several times till a preestablished convergence criteria, which measures the performance of the trained ML model, is fulfilled.

Privacy issues: FL successfully avoids having all the training data gathered by the same centralized entity, subsequently reinforcing data control and privacy protection by isolating the local datasets in each client premises. Unfortunately, even though local data is not openly shared, if no further protection measures are applied, FL protocols still present significant

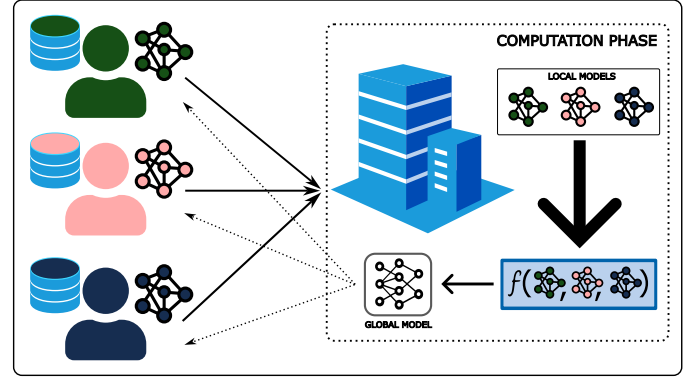


Fig. 1: High-level description of the FL protocol.

privacy and security issues [1]. Several works have shown how the exchanged clients’ local ML updates can sometimes remember information of the data used during training, thus making them susceptible to various types of inference attacks, especially when adversaries have white-box access to both the exchanged local and aggregated updates [3], [4].

By taking a closer look at the FL protocol, we observe that, even under a common security model as honest-but-curious with collusions and in contrast to the clients who only receive the aggregated ML update, the aggregator has direct access to all the individual updates sent by the clients. This additional information accessible to the aggregator aggravates the potential privacy risks posed by inference attacks, especially if the aggregator is compromised by an adversary. Consequently, this situation underlines the importance of incorporating adequate private aggregation solutions into the FL pipeline. As surveyed in [5], a diverse number of technologies can be used for this purpose, although there is no definitive perfect candidate.

One prominent example of those technologies corresponds to Homomorphic Encryption (HE) which, when extended to operate with multiple keys [6], [7], seems to precisely cover many of the requirements of private aggregation protocols.

Scope of our work: Recent research in the cryptographic community [8], [9] has shown that mainstream HE schemes paired with multiple keys can introduce some unexpected security vulnerabilities. In view of this privacy weakness, we focus here on how to correctly use modern HE schemes (BFV [10], [11] for exact arithmetic and CKKS [12] for approximate arithmetic) with multiple keys for the execution of private

average aggregation. Our main interest lies on cross-silo FL settings, in which the ML model is built from stateful clients, who are always available and have enough computational power. Even so, we note that HE could also be adapted to work reasonably well in the cross-device setting if the clients, who are intermittently available, have still enough computational and memory resources for encryption/decryption.

A. Main contributions

We propose private aggregation methods based on the use of HE under multiple keys. Our solutions follow the blueprint of [6] making use of additive secret sharing to deal with secret keys. In particular, our main contribution is threefold:

- We survey the state of the art in threshold HE and its application for implementing efficient private aggregation protocols. In particular, we explore in detail the limitations of the commonly used IND-CPA security model in characterizing the privacy vulnerabilities present in threshold HE when applied to FL protocols. We also discuss the main countermeasures to securely instantiate these schemes under the IND-CPA^D security model [13].
- Contrarily to what was previously established in [8], we show how, for the federated average rule, the CKKS scheme can still outperform BFV in terms of cipher expansion when a high smudging noise is used to guarantee privacy. We study and compare the range of parameters for which one scheme behaves better than the other.
- We briefly discuss how, for the FL setting, the approximate nature of CKKS not only does not introduce additional cryptographic weaknesses as stated in [8], but also can bring some extra privacy protection benefits by naturally adding noise to the aggregated result.

B. Notation and structure

Polynomials are denoted with regular lowercase letters, ignoring the polynomial variable (e.g., a instead of $a(x)$) whenever there is no ambiguity. We also represent polynomials as column vectors of their coefficients $\vec{a}$. Let $R_q = \mathbb{Z}[x]/(1+x^n)$, with $q \in \mathbb{N}$ and n a power-of-two, be the polynomial ring in the variable x reduced modulo $1+x^n$ with coefficients belonging to $\mathbb{Z}_q$, but using the set of representatives $(-q/2, q/2]$. Let χ be the error distribution over R_q , whose coefficients are independently sampled from a discrete Gaussian with standard deviation σ and truncated support over $[-B, B]$. Then, for a finite set $\mathcal{S}$, $x \leftarrow \mathcal{S}$ denotes sampling x uniformly at random from $\mathcal{S}$, while $x \leftarrow \chi$ denotes sampling from the distribution χ . Finally, both $\|\vec{a}\|$ and $\|a\|$ refer to the infinity norm of $\|\vec{a}\|$.

The rest of the document is organized as follows: Section II briefly reviews the state of the art of HE and its application for private aggregation. We pay special attention to the privacy vulnerabilities present in threshold HE when the adversary has access to a restricted decryption oracle. Section III details the baseline BFV and CKKS schemes. It also introduces threshold HE as a natural extension dealing with multiple keys inside the FL setting. We focus on describing the use of the smudging noise as the main countermeasure against

the attacks discussed in [8], [9]. We study its effects and we provide analytical comparisons among both threshold BFV and threshold CKKS. We also discuss possible advantages that the approximate nature of CKKS can bring into the FL setting. Finally, Section IV concludes with some future work lines.

II. PRELIMINARIES: STATE OF THE ART FOR HE

Even though the application of HE for private aggregation has already been demonstrated in several works [14]–[17], the use of more modern lattice-based HE schemes, such as BFV, BGV, and CKKS [18], is more recent. Specifically, these schemes are based on the hardness of the Ring Learning with Errors (RLWE) problem and offer significant efficiency advantages by naturally supporting SIMD (Single-Instruction Multiple-Data) operations, which allow packing several values together under the same ciphertext [19]. These features have motivated research into the combination of RLWE-based HE with other complementary privacy-related tools. For instance, some works explore their integration with verification techniques to address malicious servers [20], [21], and with Differential Privacy (DP) to better manage a larger number of colluding dishonest clients in FL [22], among others.

A. Limitations for FL deployment with HE

An advantage of HE-based aggregations is that they naturally align with the communication flow of the described FL protocol between clients and the aggregator. First, the clients encrypt their updates, then the aggregator homomorphically computes the aggregation function $f_{\Sigma}(\vec{w}^1, \dots, \vec{w}^L)$, and finally, the clients can decrypt the aggregated update. However, these works assume a very simple deployment scenario in which all clients share the same secret and public keys, implicitly imposing a strong non-colluding assumption among the aggregator and those parties with access to the secret key.

HE and multiple keys: To overcome this limitation, we can upgrade HE schemes to handle ciphertexts under multiple keys [7]. Several works propose methods to homomorphically aggregate ciphertexts encrypted under different keys [23], [24], but the most versatile option corresponds to threshold variants of single-key HE, such as the MHE (Multiparty HE) scheme proposed in [6]. In this scheme, the secret key associated with the public key of a single-key HE scheme is divided into several additive shares, which are distributed among the clients. The secret shares are generated during a setup phase that is run offline before starting the FL protocol. Note that MHE enables encryption of the local updates under a common secret key that is unknown to any single party, requiring decryption to be done collaboratively by all clients.

In return for the need for a setup phase, threshold HE schemes scale very well with the number of clients and allow for easy reuse of results from single-key HE. For example, in [23], [24], efficient homomorphic aggregations are limited to average aggregation. In contrast, with threshold schemes, we could utilize the results of [25], where the authors studied the homomorphic evaluation of several Byzantine-robust aggregation rules (e.g., the median). Unfortunately,

in [8], [9], the authors demonstrate how threshold HE schemes are particularly vulnerable when there is a non-negligible probability of incorrect decryption. Both works exemplify how this vulnerability can be exploited to implement effective key-recovery attacks against several mainstream HE libraries.

B. Security models for HE: From IND-CPA to IND-CPA^D

Previous works [6] using threshold HE discussed how these schemes could be proven to be secure under the IND-CPA (Indistinguishability under Chosen-Plaintext Attack) security model by relying on the difficulty of breaking the RLWE problem. Under this passive security model, the adversary only has access to encryptions by means of an encryption oracle. Note that this security model is sufficient from the aggregator's perspective, who only sees RLWE ciphertexts and the public key. However, the absence of a decryption oracle does not fully capture the peculiarities of the aggregation functionality, especially from the clients' perspective, who can collaboratively decrypt ciphertexts without knowing the secret key. The consequences of having a decryption oracle for HE schemes is precisely an issue which is being explored since the introduction of the IND-CPA^D security model in [13], [26].

III. AN ANALYSIS OF PRIVATE AGGREGATION WITH HE

This section surveys the available algorithms to homomorphically evaluate $f_{\Sigma}(\cdot)$ in an FL protocol. We start by giving a brief overview of how HE fits inside the FL protocol. Next, we introduce additive HE and describe how a threshold variant can be built on top of the single-key counterparts. We then focus on the IND-CPA^D security model and the use of smudging noise as a countermeasure against the key-recovery attacks proposed in [8], [9]. Finally, we analyze the performance degradation caused by the large smudging noise, and provide analytical comparisons among different solutions for private aggregation based on threshold BFV and threshold CKKS. We indicate for which protocol parameters one scheme outperforms the other.

General Overview: The incorporation of HE to securely compute the aggregation function inside the FL protocol obeys the following two principles: (1) unprotected local data is isolated in different silos and remains in the clients' premises, and (2) all local updates leaving a silo are previously encrypted with HE. This work follows the blueprint of the threshold Multiparty Homomorphic Encryption (MHE) scheme proposed in [6] (see Section III-B for further details). Figure 2 sketches the process followed to train a model. Note that steps 2-5 are typically repeated several times till convergence is achieved:

- 1) *Setup step:* All involved clients generate individual secret and a collective public keys. An initial common ML model is agreed among the Clients and the Aggregator.
- 2) *Local training step:* Each Client trains locally with its data a local ML update.
- 3) *Input step:* Each Client encrypts its local ML update to be outsourced under a collective public key.
- 4) *Evaluation step:* The aggregator will be in charge of homomorphically computing the functionality f_{Σ} .

- 5) *Output step:* Finally, all involved parties collaboratively decrypt the aggregated ML update.

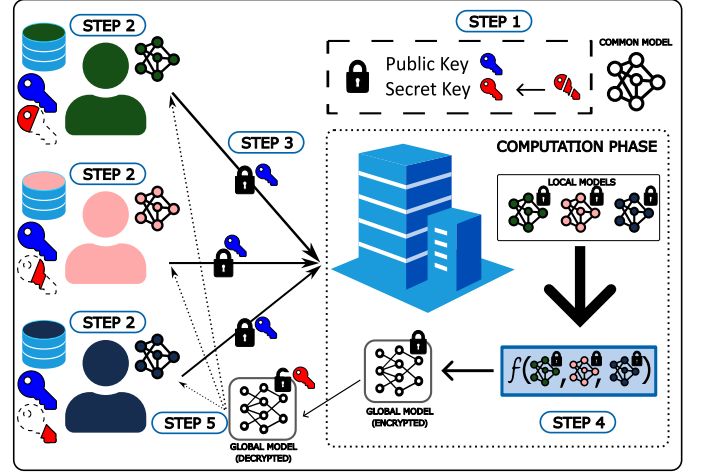


Fig. 2: Protocol for private average aggregation.

A. Background: Single-Key HE

As our interest in this work relies on the computation of average aggregations $\vec{w}^* = f_{\Sigma}(\vec{w}^1, \dots, \vec{w}^L)$, we will restrict ourselves to additive HE schemes. For ease of exposition we will focus on the case of addition $f_{\Sigma}(\vec{w}^1, \dots, \vec{w}^L) = \sum_i \vec{w}^i$.

Definition 1 (Additive Homomorphic Encryption). Let $E = \{\text{Setup}, \text{SecKeyGen}, \text{PubKeyGen}, \text{Enc}, \text{Dec}\}$ be an asymmetric encryption scheme, whose security is parameterized by λ . An additive homomorphic encryption scheme extends E with the Add procedure, having $E_{\text{ahe}} = \{E.\text{Setup}, E.\text{SecKeyGen}, E.\text{PubKeyGen}, E.\text{Enc}, E.\text{Dec}, \text{Add}\}$, where:

- Let $c_{\vec{v}}$ and $c_{\vec{w}}$ be two ciphertexts encrypting, respectively, $\vec{v}$ and $\vec{w}$. The procedure Add satisfies:

$$E_{\text{ahe}}.\text{Dec}(E_{\text{ahe}}.\text{Add}(c_{\vec{v}}, c_{\vec{w}})) = \vec{v} + \vec{w}. \quad (1)$$

We have chosen two *state-of-the-art* RLWE-based cryptosystems to instantiate the E_{ahe} scheme which is informally introduced in Definition 1. Specifically, they correspond to BFV [10], [11] and CKKS [12], being each of them, respectively, a representative of an exact and an approximate HE scheme. We refer the reader to Table I for more details on their primitives and parameters.

Both schemes allow us to perform SIMD-style additions by packing a total of n values.¹ However, they still present some significant differences regarding correctness [27]. While for BFV we can choose adequate parameters which satisfy Eq. (1) for all the homomorphic additions needed to compute $f_{\Sigma}(\cdot)$, capturing the behavior of approximate HE requires to relax the exact correctness of E . This can be done by considering $E_{\text{ahe}}.\text{Dec}(E_{\text{ahe}}.\text{Add}(c_{\vec{v}}, c_{\vec{w}})) \approx \vec{v} + \vec{w}$ instead. Now, given a

¹For the more general case of SIMD-style additions and multiplications, we can pack in each separate ciphertext a total of, respectively, n and $n/2$ values for BFV and CKKS.

TABLE I: Summary of BFV & CKKS HE Schemes

Parameters remarks		
BFV & CKKS: R_q is the ciphertext ring. BFV: $\Delta = \lfloor \frac{q}{t} \rfloor$. R_t is the plaintext ring, with $q > t$. CKKS: Given an arbitrary target margin error ϵ and B , we can set the scale Δ .		
Cryptographic Primitives		
$E_{\text{ahe}}.\text{Setup}() = pp$	BFV	Select $pp \leftarrow (t, n, q, \sigma, B, \Delta)$
	CKKS	Select $pp \leftarrow (\epsilon, n, q, \sigma, B, \Delta)$
$E_{\text{ahe}}.\text{SecKeyGen}(1^\lambda) = \text{sk}$	Given the security parameter λ , sample $s \leftarrow R_3$. Output $\text{sk} = s$	
$E_{\text{ahe}}.\text{PubKeyGen}(\text{sk}) = \text{pk}$	Sample $p_1 \leftarrow R_q$, and $e \leftarrow \chi$. $\text{pk} = (p_0, p_1) = (-\text{sk} \cdot p_1 + e, p_1)$	
$E_{\text{ahe}}.\text{Enc}(\text{pk}, m) = \text{ct}$	Sample $u \leftarrow R_3$ and $e_0, e_1 \leftarrow \chi$. $\text{ct} = (c_0, c_1) = (\Delta m + u \cdot p_0 + e_0, u \cdot p_1 + e_1)$	
$E_{\text{ahe}}.\text{Add}(\text{ct}, \text{ct}') = \text{ct}_{\text{add}}$	Given $\text{ct} = (c_0, c_1)$ and $\text{ct}' = (c'_0, c'_1)$. $\text{ct}_{\text{add}} = (c_0 + c'_0, c_1 + c'_1)$	
$E_{\text{ahe}}.\text{Dec}(\text{sk}, \text{ct}) = m$	BFV	$m = \lfloor \frac{t}{q} [c_0 + c_1 \cdot s]_q \rfloor_t$
	CKKS	$m \approx \frac{[c_0 + c_1 \cdot s]_q}{\Delta}$
Decryption remarks		
BFV: Decryption correctness holds if $(2n+1)B < \frac{q}{2t} - \frac{t}{2}$ (see Lemma 1), being $n > \delta_R$ an upper bound for the expansion factor δ_R of the ring R [28]. CKKS: After decryption, we obtain $\tilde{m} = m + e_{\text{ct}} \approx m$, where e_{ct} is the internal error of the ciphertext, with $\ e_{\text{ct}}\ < \frac{(2n+1) \cdot B}{\Delta}$.		

certain error margin $\epsilon > 0$, we say that the CKKS scheme is approximately correct, if the following holds:

$$\|f(\vec{w}^1, \dots, \vec{w}^L) - E_{\text{ahe}}.\text{Dec}(f(c_{\vec{w}^1}, \dots, c_{\vec{w}^L}))\| < \epsilon.$$

Some details on the HE schemes: To ensure a fair comparison between BFV and CKKS, we have made some minor modifications to the original CKKS scheme described in [12]. These changes primarily involve sampling the randomness in both BFV and CKKS from the same distributions. Additionally, CKKS and BFV make use of different slot packing methods [12], [19], which produce some variations in the operations performed before and after encryption and decryption. However, because our interest in this work lies in homomorphic additions, for which SIMD-type operations hold without packing, we have removed this step in both schemes. Consequently, some of the procedures described in Table I become identical for both schemes.

B. A Protocol for Private Aggregation with Multiparty HE

To securely evaluate the aggregation functionality $f_\Sigma(\cdot)$, we resort to the extension of the single-key HE schemes from Section III-A into their threshold variants. In particular, we make use of the Multiparty Homomorphic Encryption (MHE) scheme introduced in [6] as a solution to the following *multiparty aggregation problem*:

Definition 2 (Adapted from Def. 1 in [29] to our FL setting). Let $\mathcal{C} = \{C_1, C_2, \dots, C_L\}$ be a set of L clients, where each client C_i holds an input $\vec{w}^i$ (*input and receiver parties*). Let $f_\Sigma(\vec{w}^1, \vec{w}^2, \dots, \vec{w}^L) = \vec{w}^*$ be the average aggregation function (f_Σ is called *ideal functionality*) over the *input parties*. Let

TABLE II: Multiparty Private Aggregation Protocol

Public input: ideal aggregation functionality f_Σ to be computed Private input: $\vec{w}^i$ for each $C_i \in \mathcal{C}$ Output for all clients in $\mathcal{C}$: $\vec{w}^* = f(\vec{w}^1, \vec{w}^2, \dots, \vec{w}^L)$	
Setup	All clients C_i instantiate the multiparty homomorphic scheme E_{mhe} $\text{sk}_i = E_{\text{mhe}}.\pi_{\text{SecKeyGen}}(\lambda, \kappa)^*$ $\text{cpk} = E_{\text{mhe}}.\pi_{\text{PubKeyGen}}(\kappa, \text{sk}_1, \dots, \text{sk}_L)$
Input	Each C_i encrypts its input $\vec{w}^i$ and provides it to the <i>Aggregator</i> ⁺ $c_{\vec{w}^i} = E_{\text{mhe}}.\text{Enc}(\text{cpk}, \vec{w}^i)$
Evaluation	The <i>Aggregator</i> computes the encrypted output for the ideal functionality f_Σ relying on $E_{\text{mhe}}.\text{Add}$ $c_{\vec{w}^*} = f_\Sigma(c_{\vec{w}^1}, c_{\vec{w}^2}, \dots, c_{\vec{w}^L})$
Output	The parties in $\mathcal{C}$ execute the decryption protocol $\vec{w}^* = E_{\text{mhe}}.\pi_{\text{Dec}}(\text{sk}_1, \dots, \text{sk}_L, c_{\vec{w}^*})$

* κ parameterizes the homomorphic capacity of the scheme E_{Add} .
+ The *Aggregator* is in charge of performing the aggregation.

$\mathcal{A}$ be a static semi-honest adversary that can corrupt up to $L-1$ clients in $\mathcal{C}$ and let $\mathcal{C}_\mathcal{A}$ be the subset of clients corrupted by $\mathcal{A}$. Then, the *secure multiparty aggregation problem* consists in providing $\mathcal{C}$ with $\vec{w}^*$, yet $\mathcal{A}$ must learn nothing more about $\{\vec{w}^i\}_{C_i \notin \mathcal{C}_\mathcal{A}}$ than what can be deduced from the inputs $\{\vec{w}^i\}_{C_i \in \mathcal{C}_\mathcal{A}}$ and output $\vec{w}^*$ it controls (this property is called *input privacy*).

Consequently, a solution to the *secure multiparty aggregation problem* consists in a protocol π_{f_Σ} which realizes the *ideal functionality* f_Σ while also preserving the *input privacy* [30].

We can make use of the general MHE-based solution proposed in [6], which is proved on the Common Reference String (CRS) model,² and it also assumes that all parties are connected through authenticated channels. The description of our particularized protocol for private aggregation is described in Table II. It relies on the existence of the following augmented multiparty encryption scheme:

Definition 3 (Multiparty Additive HE). Let E_{ahe} be the asymmetric and additive homomorphic encryption scheme from Definition 1, whose security is parameterized by λ (see Table I for two concrete examples with BFV and CKKS), and let $S = (S.\text{Share}, S.\text{Combine})$ be an L -party secret sharing scheme. The associated *multiparty homomorphic encryption scheme* (E_{mhe}) is obtained by applying the secret-sharing scheme S to E_{ahe} 's secret key sk (*ideal secret key*) and is defined as the tuple $E_{\text{mhe}} = (E_{\text{ahe}}.\text{Enc}, E_{\text{ahe}}.\text{Dec}, E_{\text{ahe}}.\text{Add}, E_{\text{ahe}}^S)$, where $E_{\text{ahe}}^S = (\pi_{\text{SecKeyGen}}, \pi_{\text{PubKeyGen}}, \pi_{\text{Dec}})$ is a set of multiparty protocols executed among the clients in the set $\mathcal{C}$, and having the following *private ideal functionalities* for each client C_i :

- *Ideal secret-key generation*:
 $f_{i, \pi_{\text{SecKeyGen}}}(\lambda) = S.\text{Share}_i(E_{\text{ahe}}.\text{SecKeyGen}(\lambda)) = \text{sk}_i$.
- *Collective public-key generation*:
 $f_{\pi_{\text{PubKeyGen}}}(\text{sk}_1, \text{sk}_2, \dots, \text{sk}_L)$
 $= E_{\text{ahe}}.\text{PubKeyGen}(S.\text{Combine}(\text{sk}_1, \text{sk}_2, \dots, \text{sk}_L))$.
- *Collective decryption*:

²It assumes that all parties have access to a common random string.

$$f_{\pi_{\text{Dec}}}(\text{sk}_1, \text{sk}_2, \dots, \text{sk}_L, \text{ct}) \\ = E_{\text{ahe}}.\text{Dec}(\text{S.Combine}(\text{sk}_1, \text{sk}_2, \dots, \text{sk}_L), \text{ct}).$$

Definition 3 particularizes Def. 2 from [29] to the case in which the encryption scheme E_{ahe} is additive homomorphic.

Concrete instantiations for Multiparty HE: The private ideal functionalities introduced in Definition 3 can be implemented by the clients with concrete protocols:

- $f_{i, \pi_{\text{SecKeyGen}}}(\lambda)$: Each client C_i independently runs the procedure $E_{\text{Add}}.\text{SecKeyGen}(1^\lambda) = \text{sk}_i$.
- $f_{\pi_{\text{PubKeyGen}}}(\text{sk}_1, \text{sk}_2, \dots, \text{sk}_L)$: Given a common random polynomial p_1 , each client $C_i \in \mathcal{C}$ samples $e_{\text{pk},i} \leftarrow \chi$ and discloses to the other clients $p_{0,i} = -p_1 \cdot \text{sk}_i + e_{\text{pk},i}$. The collective public key is computed as:

$$\text{cpk} = \left(\sum_{C_i \in \mathcal{C}} p_{0,i}, p_1 \right) = \left(-p_1 \underbrace{\sum_i \text{sk}_i}_{\text{sk}} + \sum_i e_{\text{pk},i}, p_1 \right).$$

- $f_{\pi_{\text{Dec}}}(\text{sk}_1, \text{sk}_2, \dots, \text{sk}_L, \text{ct})$: The corresponding collaborative decryption protocol can be divided in two phases.
 - Given a ciphertext $\text{ct} = (c_0, c_1)$ encrypted under the ideal secret-key sk , each client computes its partial decryption of ct . Each client C_i samples $e_{\text{smg},i} \leftarrow \chi$ and discloses:

$$h_i = \text{sk}_i \cdot c_1 + e_{\text{smg},i}.$$

- Given all the decryption shares h_i from the clients, the protocol outputs:

$$d = \left[c_0 + \sum_{C_i \in \mathcal{C}} h_i \right]_q \\ = \Delta m + \underbrace{e \cdot v + e_0 + s \cdot e_1}_{e_{\text{ct}}} + \underbrace{\sum_i e_{\text{smg},i}}_{e_{\text{smg}}}.$$

The final step for decryption depends on whether we use BFV or CKKS as the baseline single-key HE scheme:

- For BFV, the obtained decryption is $m = \lfloor \frac{t}{q} d \rfloor_t$.
- For CKKS, the obtained decryption is $m + \frac{e_{\text{ct}} + e_{\text{smg}}}{\Delta} = \frac{d}{\Delta}$.

We can obtain lower bounds for q , similar to the single-key counterparts, by taking into account the larger noise terms present in the threshold versions (see Section III-D).

C. Multiparty HE with Countermeasures for IND-CPA^D

The new notion IND-CPA^D extends IND-CPA by allowing the adversary to have access to a very restricted decryption oracle which can only be used for genuine ciphertexts, or ciphertexts obtained from genuine ciphertexts through valid homomorphic operations. By adapting the definition to our specific case with the aggregation function f_Σ , the idea is that an adversary knowing $\{\vec{w}^1, \dots, \vec{w}^L\}$ and f_Σ , can also obtain $f_\Sigma(\vec{w}^1, \dots, \vec{w}^L)$. Then, if this adversary has access to $\{\text{Enc}(\vec{w}^1), \dots, \text{Enc}(\vec{w}^L)\}$ and homomorphically evaluates the aggregation function f_Σ as $f_\Sigma(\text{Enc}(\vec{w}^1), \dots, \text{Enc}(\vec{w}^L))$, then she should not gain more information from $\text{Dec}(f_\Sigma(\text{Enc}(\vec{w}^1), \dots, \text{Enc}(\vec{w}^L)))$ than what

she can already obtain from $f_\Sigma(\vec{w}^1, \dots, \vec{w}^L)$. Unfortunately, in [13] the authors showed how some HE schemes, which are IND-CPA secure, leak information through its difference and, hence, are not IND-CPA^D secure. An example is the case of approximate HE schemes like CKKS [12], for which the difference $\text{Dec}(\text{Enc}(\vec{w}^i)) - \vec{w}^i$ directly leaks the internal noise of the underlying RLWE sample, then allowing the adversary to break the RLWE indistinguishability assumption.

Till very recently, the cryptographic community believed that only approximate HE presented this vulnerability and that exact HE schemes like BFV [10], [11], BGV [31] or CGGI [32] were invulnerable to this type of attacks, hence being these schemes naturally safe under IND-CPA^D. However, a couple of new works [8], [9] have shown that this vulnerability could also be present in practice for exact schemes, as soon as they present a non-negligible probability of incorrect decryption. Both works exemplify how this vulnerability can be exploited to implement effective key-recovery attacks against several mainstream HE libraries.

In both works [8], [9], the authors illustrate how these types of attacks are especially relevant when dealing with threshold HE schemes. Fortunately, they also propose several countermeasures to achieve IND-CPA^D security when using threshold HE. In particular, the addition during decryption of an adequate smudging noise with λ -independent variance can do the job. It is worth highlighting that [8] showcases how this countermeasure significantly impacts the cryptosystem's parameters, subsequently reducing its efficiency. According to the authors, this effect is even more pronounced for CKKS, for which using a large-variance smudging noise is likely to severely reduce the precision of the decrypted result.

1) *Impact of the smudging noise:* In the end, given a ciphertext ct , the key-recovery attacks mentioned above aim at extracting its noise component e_{ct} . If this extraction is feasible, the secret key can be recovered via linear algebra techniques. For approximate HE schemes, this process is relatively easy once we have access to the decryption. However, for exact HE schemes, these attacks attempt to force decryption failures as a mechanism to estimate the corresponding e_{ct} component.

Therefore, even in extreme cases where the post-processed decryption can become unusable, adding a large enough smudging noise after decryption is required to hide any information leakage regarding the original error e_{ct} .

In relation to this countermeasure, recent work by [27] discusses how the *application-agnostic* nature of IND-CPA^D often leads to impractically large parameters when adding a large smudging noise. This is due to the fact that e_{smg} must statistically hide e_{ct} for all possible homomorphic circuits which satisfy correctness for the initially chosen cryptosystem parameters with the $E_{\text{ahe}}.\text{Setup}()$ procedure. The authors propose a relaxation of this definition, termed *application-aware* IND-CPA^D, which is more suitable for the FL setting.

In our work, we follow their guidelines and assume that both Clients and the Aggregator will behave semi-honestly, which here means that they are *expected to follow exactly the pre-*

scribed instructions to compute $c_{\vec{w}^*} = f_{\Sigma}(c_{\vec{w}^1}, c_{\vec{w}^2}, \dots, c_{\vec{w}^L})$ and encrypt $\{c_{\vec{w}^1}, c_{\vec{w}^2}, \dots, c_{\vec{w}^L}\}$ (see Tables I and II).

Collaborative decryption protocol: Given again ct, once all partial decryptions $h_i = \text{sk}_i \cdot c_1 + e_{\text{smg},i}$ are gathered, the adversary has direct access to $d = \Delta \cdot m + e_{\text{ct}} + e_{\text{smg}}$. Our objective is to obtain e_{smg} such that $e_{\text{ct}} + e_{\text{smg}}$ is statistically indistinguishable from fresh noise $\tilde{e}_{\text{smg}}$ (see the Smudging Lemma in [33]). To ensure IND-CPA^D security, we adhere to the practical guidelines provided in [8], which recommend a Gaussian smudging noise with variance $\sigma_{\text{smg}}^2 = 2^\lambda \sigma_{\text{ct}}^2$. By applying an upper-bound $B = \mathcal{O}(\sigma)$,³ this results in $B_{\text{smg}} = 2^{\frac{\lambda}{2}} B_{\text{ct}}$ (e.g., $\lambda = 128$ is suggested in [6]).

D. Comparison analysis between BFV and CKKS

In [8], the authors discuss how the fact that CKKS is particularly vulnerable to the exposure of decryptions under IND-CPA^D makes working with its threshold variant, by following the blueprint of [6] (see Section III-B), less useful compared to other exact schemes such as BFV. The main argument provided relies on the fact that the high variance of the required smudging noise is likely to jeopardize the precision of the finally decrypted results. Our aim in this section is to elaborate more precisely on this comparison in the context of FL. In particular, we compare the bit precision achieved by the threshold variants of both BFV and CKKS for the homomorphic evaluation of the average aggregation rule.

Choosing multiparty cryptosystems' parameters: We can upper-bound the noise of a fresh single-key ciphertext by $(2n+1)B$, relying on Lemma 1 (see Appendix). To extend this result to the multiparty HE schemes from Section III-B, we must consider the following: (1) The secret key and error for the cpk are L times larger, which implies that the noise of fresh ciphertexts is now upper-bounded by $B(2nL+1)$. (2) During the homomorphic execution of the f_{Σ} functionality, the underlying noise is further increased by a factor of L . (3) During the collective decryption of the resulting ct, every client adds to e_{ct} a smudging noise satisfying $\|e_{\text{smg},i}\| \leq B_{\text{smg}}$.

As a result, the final noise $e_{\text{ct}} + e_{\text{smg}}$ present in the collective decryption of ct is upper-bounded by $B_{\text{ct}}^{\text{MP}} = B_{\text{ct}} + LB_{\text{smg}} = (1 + L2^{\lambda/2})B_{\text{ct}}$, with $B_{\text{ct}} = LB(2nL+1)$. As $q = \mathcal{O}(B_{\text{ct}}^{\text{MP}})$ (see Prop. 1) for both BFV and CKKS, we can see how the smudging noise significantly impacts ciphertext size and, subsequently, the efficiency of the private aggregation protocol.

More specifically, the remarks for decryption with single-key HE given in Table I can be adapted to their threshold variants by considering $B_{\text{ct}}^{\text{MP}}$ instead of B_{ct} . This results in the expressions: (1) $B_{\text{ct}}^{\text{MP}} < \frac{q}{2t} - \frac{t}{2}$ for Multiparty BFV (MBFV) and, (2) $\Delta B_m + B_{\text{ct}}^{\text{MP}} < q/2$, with $\|m\| < B_m$ and $m = \sum_i^L m_i$, for Multiparty CKKS (MCKKS).

A comparison in terms of cipher expansion: From the previous expressions, we can obtain lower-bounds for q , which, by following Prop. 1 (see Appendix), can be used to

compare the cipher expansion of both schemes. In particular, MCKKS has a smaller q than MBFV if:

$$B_{\text{ct}}^{\text{MP}} > \frac{2\Delta_{\text{CKKS}}B_m - t^2}{2(t-1)}, \quad (2)$$

where we use Δ_{CKKS} to make explicit that we refer to the Δ from CKKS in Table I. Note that Δ_{CKKS} and t do not play the same role in MCKKS and MBFV, respectively.

To ensure a fair comparison between both schemes, we express Δ_{CKKS} in terms of the error margin $\epsilon = B_{\text{ct}}^{\text{MP}}/\Delta_{\text{CKKS}}$ introduced in Section III-A. This allows us to compare the achieved bit precision of MCKKS, represented by $\log_2(\epsilon^{-1}B_m)$, directly with the bit precision $\log_2 t$ achieved in MBFV. Also, for simplicity in the comparison, we assume that the input plaintexts used in MCKKS are normalized beforehand to ensure an aggregation satisfying $B_m < 1$. These changes lead to the following inequality:

$$\frac{t^2}{2B_{\text{ct}}^{\text{MP}}} + t - 1 > \epsilon^{-1}. \quad (3)$$

In Figures 3 and 4, we plot the expression from Eq. (3) on a logarithmic scale, using bit precisions $\log_2 t$ and $\log_2 \epsilon^{-1}$. Note that if a point $(\log_2 t, \log_2 \epsilon^{-1})$ belongs to the colored area, then the modulus q required for MCKKS is smaller than that of MBFV. In Figure 3, we set parameters to $L = 10$, $n = 8192$ and $B = 19.2$, while λ ranges over the set $\{32, 64, 96, 128\}$. In Figure 4, we fix the parameters at $n = 8192$, $B = 19.2$, and $\lambda = 128$, while L takes values in $\{8, 128\}$. The parameter values are practical enough for the cross-silo setting, and ensure a bit security of at least 128 for all the represented bit precisions.

We observe that while increasing λ and L up to 128 does reduce the range of parameters for which MCKKS outperforms MBFV (the effect is more significant when increasing λ), the approximate MCKKS compares similarly with MBFV, getting better for very high bit precisions. Finally, by taking a closer look to the figures, we can also see how (3) approximately follows a piecewise linear function with two intervals:

- In the first interval, the addend $t - 1$ dominates the left part of Eq. (3), allowing us to simplify the inequality to approximately $\log_2(t-1) > \log_2 \epsilon^{-1}$. Here, both schemes behave similarly for the same bit precision.
- In the second interval, the addend $\frac{t^2}{2B_{\text{ct}}^{\text{MP}}}$ dominates, simplifying the inequality into approximately $2\log_2 t - \log_2 B_{\text{ct}}^{\text{MP}} - 1 > \log_2 \epsilon^{-1}$. In this case, the bit precision of MCKKS grows twice as fast as MBFV.

Runtime comparisons: The implementation runtimes presented in this section were conducted single-threaded on an Intel Core i7-10750H CPU @ 2.60GHz $\times$ 12 with 31.1 GB. We considered two aggregation protocols with bit precision $\log_2 t$ and $\log_2 \epsilon^{-1}$ for MBFV and MCKKS, respectively. Table III includes three different parameter sets, and Table IV presents their corresponding implementation runtimes in Lattigo [6].

Further clarifications on approximate HE: There are a few points that we must clarify regarding our comparison.

³ $B = 6\sigma$ is commonly used by HE libraries.

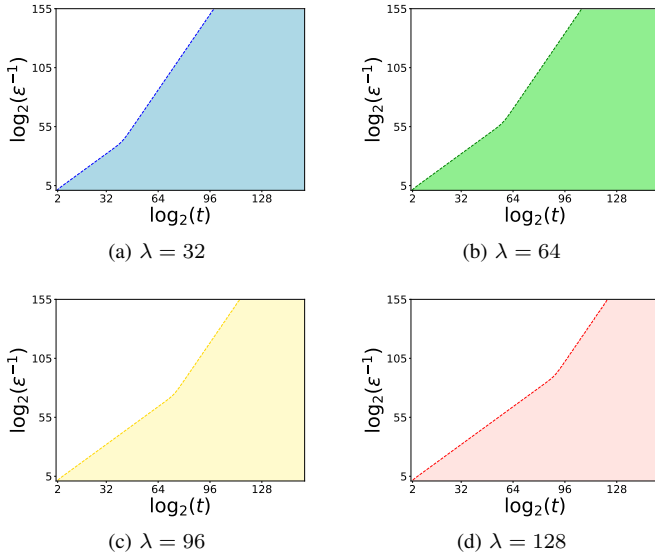


Fig. 3: Comparison of q for MBFV and MCKKS varying λ .

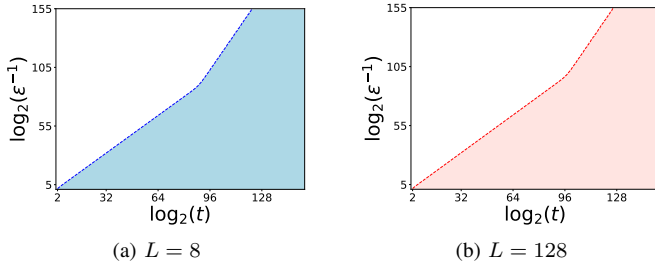


Fig. 4: Comparison of q for MBFV and MCKKS varying L .

Param.	Set 1 BFV&CKKS	Set 2 BFV	Set 3 CKKS
$\{n, L\}$	$\{16384, 16\}$	$\{16384, 32\}$	$\{16384, 32\}$
$\{t, \epsilon^{-1}\}$ [bits]	$\{45, 45\}$	$\{60, -\}$	$\{-, 60\}$
$\{\#Limbs, q\}$ [bits]	$\{4, 240\}$	$\{10, 300\}$	$\{9, 270\}$
$\{q_{BFV}, q_{CKKS}\}$ [bits]	$\{232, 238\}$	$\{280, -\}$	$\{-, 259\}$

TABLE III: Example parameter sets for the private aggregation protocol ($\sigma = 3.2$, $\lambda = 128$ and bit security = 128).

Protocol step	Par. set 1	Par. set 2	Par. set 3
Col. Key Gen.	Client: 4.4 ms	Client: 10.3 ms	Client: 10.1 ms
Encryption	Client: 1.1 s	Client: 2.3 s	Client: 2.1 s
Aggregation	Agg: 244.3 ms	Agg: 1.2 s	Agg: 1.1 s
Col. Dec.	Client: 2.5 s	Client: 5.2 s	Client: 4.8 s
Total runtime	3.9 s	8.7 s	8 s

TABLE IV: Implementation runtimes for the private aggregation protocol ($N_{ModelParameters} = 1638400$).

Firstly, we have considered a multiplication by t/q during decryption, following the BFV description as originally reported in [11]. The rounding error introduced when t does not divide q is precisely the source of the extra addend $\frac{t^2}{2B_{ct}^{MP}}$ in Eq.(3). If we instead modify the decryption to compute a division by Δ , Eq.(3) would resemble the left interval of Figures 3 and 4.

Consequently, ϵ^{-1} in MCKKS grows similarly to $\log_2 t$ for this modified MBFV. Still, CKKS presents a significant advantage when used for private aggregation because it nat-

urally introduces noise into the decrypted result. In contrast, exact HE protects the privacy of clients' inputs during homomorphic computation, but does not provide any additional privacy guarantees on the decrypted aggregated models against inference attacks [5]. In fact, one possibility could be to incorporate DP along with the exact HE computation [22]. This can be done by homomorphically adding noise into the encryptions, which requires saving part of the $\log_2 t$ available bits. Instead, following our previous analysis, the bit precision ϵ^{-1} considered in CKKS does not suffer any change as the effect of the noise is already contemplated in its definition.

The question of whether the noise present in CKKS could be reused to provide theoretical DP guarantees has been very recently explored in [34]. The authors are able to provide a certain privacy budget for the harder case in which the homomorphic evaluation makes the internal noise dependent on the input messages. Note that for the average aggregation rule used in this work, the situation is simpler as the noise added with CKKS is independent of the input local updates.

IV. CONCLUSIONS AND FUTURE WORK

This work surveys the use of threshold variants of exact and approximate RLWE-based HE for the efficient implementation of private average aggregation. While these types of schemes seem to be a perfect fit for executing the aggregation primitive in FL protocols, recent work has demonstrated the existence of some unexpected security vulnerabilities under the IND-CPA^D security model if these HE schemes are not correctly instantiated. We detail the use of smudging noise with a large variance as the main defense proposed by recent works and discuss its impact on performance. We provide an exhaustive analysis comparing the effective bit precision that can be achieved when applying this defense for two concrete threshold variants of HE schemes, such as BFV and CKKS. Our analysis indicates that CKKS-based average aggregations compare well with BFV-based solutions. Moreover, we discuss how the approximate nature of CKKS can provide additional privacy guarantees for the decrypted aggregated models against inference attacks. As future work, we intend to explore the use of more complex aggregation rules, optimize the variance required for the smudging noise by taking into account the peculiarities of the FL scenario compared to the more general case of IND-CPA^D, and provide a more exhaustive performance comparison between both schemes by deploying them within a practical FL use case.

ACKNOWLEDGMENT

GPSC is partially supported by the European Union's Horizon Europe Framework Programme for Research and Innovation Action under project TRUMPET (proj. no. 101070038), by FEDER and Xunta de Galicia under project "Grupos de Referencia Competitiva" (ED431C 2021/47), by FEDER and MCIN/AEI under project FELDSPAR (TED2021-130624B-C21), and by the "NextGenerationEU/PRTR" under TRUFLES and a Margarita Salas grant of the Universidade de Vigo.

REFERENCES

- [1] B. McMahan *et al.*, “Communication-efficient learning of deep networks from decentralized data,” in *AISTATS*, ser. PMLR, vol. 54. PMLR, 2017, pp. 1273–1282.
- [2] P. Kairouz *et al.*, “Advances and open problems in federated learning,” *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021.
- [3] M. Nasr *et al.*, “Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning,” in *SP*. IEEE, 2019, pp. 739–753.
- [4] K. Leino and M. Fredrikson, “Stolen memories: Leveraging model memorization for calibrated white-box membership inference,” in *USENIX*. USENIX Association, 2020, pp. 1605–1622.
- [5] M. Mansouri *et al.*, “Sok: Secure aggregation based on cryptographic schemes for federated learning,” *Proc. Priv. Enhancing Technol.*, vol. 2023, no. 1, pp. 140–157, 2023.
- [6] C. Mouchet *et al.*, “Multiparty homomorphic encryption from ring-learning-with-errors,” *Proc. Priv. Enhancing Technol.*, vol. 2021, no. 4, pp. 291–311, 2021.
- [7] A. Aloufi *et al.*, “Computing blindfolded on data homomorphically encrypted under multiple keys: A survey,” *ACM Comput. Surv.*, vol. 54, no. 9, pp. 195:1–195:37, 2022.
- [8] M. Checri *et al.*, “On the practical CPAD security of “exact” and threshold FHE schemes and libraries,” *IACR Cryptol. ePrint Arch.*, p. 116, 2024.
- [9] J. H. Cheon *et al.*, “Attacks against the INDCPA-D security of exact FHE schemes,” *IACR Cryptol. ePrint Arch.*, p. 127, 2024.
- [10] Z. Brakerski, “Fully homomorphic encryption without modulus switching from classical gapsvp,” in *CRYPTO*, ser. LNCS, vol. 7417. Springer, 2012, pp. 868–886.
- [11] J. Fan and F. Vercauteren, “Somewhat practical fully homomorphic encryption,” *IACR Cryptol. ePrint Arch.*, p. 144, 2012.
- [12] J. H. Cheon *et al.*, “Homomorphic encryption for arithmetic of approximate numbers,” in *ASIACRYPT*, ser. LNCS, vol. 10624. Springer, 2017, pp. 409–437.
- [13] B. Li and D. Micciancio, “On the security of homomorphic encryption on approximate numbers,” in *EUROCRYPT*, ser. LNCS, vol. 12696. Springer, 2021, pp. 648–677.
- [14] E. Shi *et al.*, “Privacy-preserving aggregation of time-series data,” in *NDSS*. The Internet Society, 2011.
- [15] F. Benhamouda *et al.*, “A new framework for privacy-preserving aggregation of time-series data,” *ACM Trans. Inf. Syst. Secur.*, vol. 18, no. 3, pp. 10:1–10:21, 2016.
- [16] C. Zhang *et al.*, “Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning,” in *USENIX*. USENIX Association, 2020, pp. 493–506.
- [17] A. Madi *et al.*, “A secure federated learning framework using homomorphic encryption and verifiable computing,” in *RDAAPS*, 2021, pp. 1–8.
- [18] M. R. Albrecht *et al.*, “Homomorphic encryption standard,” *IACR Cryptol. ePrint Arch.*, p. 939, 2019.
- [19] N. P. Smart and F. Vercauteren, “Fully homomorphic SIMD operations,” *Des. Codes Cryptogr.*, vol. 71, no. 1, pp. 57–81, 2014.
- [20] D. Fiore *et al.*, “Efficiently verifiable computation on encrypted data,” in *CCS*. ACM, 2014, pp. 844–855.
- [21] D. F. Aranha *et al.*, “HELIOPOLIS: verifiable computation over homomorphically encrypted data from interactive oracle proofs is practical,” *IACR Cryptol. ePrint Arch.*, p. 1949, 2023.
- [22] A. G. Sébert *et al.*, “Combining homomorphic encryption and differential privacy in federated learning,” in *PST*. IEEE, 2023, pp. 1–7.
- [23] J. Ma *et al.*, “Privacy-preserving federated learning based on multi-key homomorphic encryption,” *Int. J. Intell. Syst.*, vol. 37, no. 9, pp. 5880–5901, 2022.
- [24] A. Pedrouzo-Ulloa *et al.*, “Practical multi-key homomorphic encryption for more flexible and efficient secure federated average aggregation,” in *IEEE CSR*. IEEE, 2023, pp. 612–617.
- [25] A. Choffrut *et al.*, “Practical homomorphic aggregation for byzantine ML,” *CoRR*, vol. abs/2309.05395, 2023.
- [26] B. Li *et al.*, “Securing approximate homomorphic encryption using differential privacy,” in *CRYPTO*, ser. LNCS, vol. 13507. Springer, 2022, pp. 560–589.
- [27] A. Alexandru *et al.*, “Application-aware approximate homomorphic encryption: Configuring FHE for practical use,” *IACR Cryptol. ePrint Arch.*, p. 203, 2024.
- [28] L. de Castro *et al.*, “Fast vector oblivious linear evaluation from ring learning with errors,” in *WAHC@CCS*, 2021, pp. 29–41.
- [29] C. Mouchet *et al.*, “Computing across trust boundaries using distributed homomorphic cryptography,” *IACR Cryptol. ePrint Arch.*, p. 961, 2019.
- [30] Y. Lindell, “How to Simulate It - A Tutorial on the Simulation Proof Technique,” in *Tutorials on the Foundations of Cryptography*. Springer, 2017, pp. 277–346.
- [31] Z. Brakerski *et al.*, “(leveled) fully homomorphic encryption without bootstrapping,” *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 13:1–13:36, 2014.
- [32] I. Chillotti *et al.*, “Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds,” in *ASIACRYPT*, ser. LNCS, vol. 10031, 2016, pp. 3–33.
- [33] G. Asharov *et al.*, “Multiparty computation with low communication, computation and interaction via threshold fhe,” in *EUROCRYPT*. Springer, 2012, pp. 483–501.
- [34] T. Ogilvie, “Differential privacy for free? harnessing the noise in approximate homomorphic encryption,” in *CT-RSA*, ser. LNCS, vol. 14643. Springer, 2024, pp. 292–315.

APPENDIX

Lemma 1. We follow the notation for BFV indicated in Table I, and assume that any $e \leftarrow \chi$ satisfies $\|e\| \leq B$. For a fresh ciphertext $ct = (c_0, c_1)$, we have $[c_0 + c_1 \cdot s]_q = \Delta m + e_{ct}$ with $\|e_{ct}\| \leq (2n+1)B$. This implies that whenever $(2n+1)B < \frac{t}{2t} - \frac{t}{2}$, decryption works correctly.

Proof. We start by computing $[c_0 + c_1 \cdot s]_q = \Delta m + \underbrace{u \cdot e + e_0 + e_1 \cdot s}_{e_{ct}}$, from which we can directly upper-bound the error polynomial e_{ct} as:

$$\|e_{ct}\| = \|u \cdot e + e_0 + e_1 \cdot s\| \quad (4)$$

$$\leq \|u \cdot e\| + \|e_0\| + \|e_1 \cdot s\| \quad (5)$$

$$\leq \delta_R B + B + \delta_R B \quad (6)$$

$$\leq nB + B + nB \quad (7)$$

$$= (2n+1)B. \quad (8)$$

Now, the decryption process requires to multiply by $\frac{t}{q}$ and apply a final coefficient-wise rounding:

$$\left\lfloor \frac{t}{q} [c_0 + c_1 \cdot s]_q \right\rfloor = \left\lfloor \frac{t}{q} (\Delta m + e_{ct}) \right\rfloor,$$

which must output m for correct decryption. If we define $\Delta = \frac{q}{t} - r$ with $0 \leq r < 1$, the condition for correct decryption can be expressed equivalently as:

$$\left\| \frac{t}{q} (-rm + e_{ct}) \right\| < \frac{1}{2}.$$

By upper-bounding the expression in the left, we obtain:

$$\left\| \frac{t}{q} (-rm + e_{ct}) \right\| \leq \frac{t}{q} (\| -rm \| + \| e_{ct} \|) < \frac{t}{q} \left(\frac{t}{2} + \| e_{ct} \| \right) < 1/2. \quad (9)$$

Therefore, combining the expressions (8) and (9), we see that decryption correctness holds if $(2n+1)B < \frac{q}{2t} - \frac{t}{2}$. $\square$

Proposition 1. Following the descriptions of the primitives from Tables I and II, the ciphertext modulus q for MBFV is larger than the q for MCKKS if the next condition is satisfied: $B_{ct}^{MP} > \frac{2\Delta B_m - t^2}{2(t-1)}$. In Section III-D, we utilize this expression as Eq. 2 to compare the cipher expansion of both schemes.

Proof. For decryption correctness, both schemes must satisfy:

- $q_{\text{MBFV}} : B_{\text{ct}}^{\text{MP}} < \frac{q}{2t} - \frac{t}{2}$.
- $q_{\text{MCKKS}} : \Delta B_m + B_{\text{ct}}^{\text{MP}} < \frac{q}{2}$.

Comparing both expressions, we have that $q_{\text{BFV}} > q_{\text{CKKS}}$ if:

$$\begin{aligned} 2tB_{\text{ct}}^{\text{MP}} + t^2 &> 2\Delta B_m + 2B_{\text{ct}}^{\text{MP}} \\ 2(t-1)B_{\text{ct}}^{\text{MP}} &> 2\Delta B_m - t^2 \\ B_{\text{ct}}^{\text{MP}} &> \frac{2\Delta B_m - t^2}{2(t-1)}. \end{aligned}$$

□