On the Security and Privacy of CKKS-based Homomorphic Evaluation Protocols

Intak Hwang, Seonhong Min, Jinyeong Seo, and Yongsoo Song

Seoul National University
{intak.hwang, minsh, jinyeong.seo, y.song}@snu.ac.kr

Abstract. CKKS is a homomorphic encryption (HE) scheme that supports approximate arithmetic over complex numbers. While it is widely used in privacy-preserving machine learning (PPML) protocols, the approximate nature of the scheme makes it challenging to formally define the security guarantees of those protocols. In particular, in a sender-receiver protocol, where the sender performs homomorphic evaluation using a private circuit, characterizing the sender's privacy remains an important open problem. Moreover, there are currently no known methods for handling malicious receivers due to the absence of a zero-knowledge argument of knowledge (ZKAoK) for the CKKS scheme.
In this paper, we address these open challenges. First, we introduce a new security definition, called *differentially private homomorphic evaluation* (DPHE), to formalize sender privacy in CKKS-based protocols. Next, we present a general compilation method that transforms a plain CKKS protocol into a DPHE protocol. Finally, we construct a zero-knowledge argument of knowledge (ZKAoK) for CKKS to achieve the DPHE property in the presence of malicious receivers, and provide concrete benchmarks of our ZKAoK implementation.
To the best of our knowledge, this is the first work to formally address security and privacy issues in CKKS-based protocols through the lens of differential privacy. We also remark that our ZKAoK is the first construction to ensure the well-formedness of CKKS public keys and ciphertexts.

Keywords: homomorphic encryption, zero-knowledge, differential privacy, CKKS

1 Introduction

Homomorphic encryption (HE) is an encryption scheme that allows computation on encrypted data without the need for decryption. Since Gentry's seminal work [Gen09], various HE schemes have been proposed, and the currently most efficient schemes [BGV12,Bra12,FV12,CGGI20,CKKS17] are based on the hardness of the Learning with Errors (LWE) problem [Reg09] or its ring variant, RLWE [LPR10]. In (R)LWE-based HE schemes, plaintexts are encrypted together with small noises, and decryption recovers both the plaintext and the noise. These (R)LWE-based HE schemes can be categorized by the types of homomorphic operations they support: BGV [BGV12] and BFV [Bra12, FV12]

support modular arithmetic over integers, TFHE [CGGI20] enables look-up table evaluation over short bit strings, and CKKS [CKKS17] supports approximate arithmetic over complex numbers.

Homomorphic Evaluation. One direct application of HE is a secure function evaluation protocol in the sender-receiver model. In this protocol, the sender possesses a circuit, while the receiver holds an input for that circuit. The objective is for the receiver to obtain the result of the circuit evaluation without revealing its input to the sender. In the semi-honest security model, HE provides a straightforward two-round solution, which we refer to as a *homomorphic evaluation* protocol. In the first round, the receiver encrypts its input using a homomorphic encryption scheme and sends the ciphertext along with the public key to the sender. The sender then evaluates the circuit on the encrypted data using homomorphic operations and returns the resulting ciphertext to the receiver. Finally, the receiver decrypts the output ciphertext to obtain the evaluation result. This construction is not only round-optimal but also communication-efficient, especially when the size of the sender's circuit significantly exceeds the size of the receiver's input. Most applications of HE, such as private information retrieval (PIR) [ACLS18, MW22], private set intersection (PSI) [CHLR18, CMdG+21], oblivious pseudorandom functions (OPRF) [ADDG24], and privacy-preserving machine learning inference (PPML) [LLL+22, JPK+24, BCCW19], fall into this scenario.

Circuit Privacy. Among applications of HE, there are cases where the sender's circuit is private. For example, in the case of PSI, the sender's circuit contains information about the elements of the sender's set; in the case of OPRF, it contains information about the secret key of the PRF; and in the case of PPML, it contains information on private training data. Unfortunately, HE alone does not guarantee the privacy of the sender's circuit, as the output ciphertext may contain information about it. This problem is known as *circuit privacy* and has been an important challenge in HE-based secure evaluation protocols since Gentry's initial construction of HE [Gen09]. In the case of semi-honest receivers, circuit privacy is typically achieved through the noise flooding method [AJLA+12, DDEK+23, Gen09], where the sender eliminates any residual circuit information from the output ciphertext by homomorphically adding a fresh encryption of zero along with exponentially large noise. The encryption of zero removes any remaining sender information from the components of the output ciphertext, while the large noise ensures that no information about the sender's circuit is retained in the ciphertext's noise, thereby revealing only the output plaintext to the receiver. This approach works effectively for exact HE schemes such as BGV, BFV, and TFHE, because, after decryption, the plaintext and noise in a ciphertext are clearly separated.

Malicious Receiver & ZKAoK. If we extend the security model to handle a malicious receiver, the sender must also verify the well-formedness of the receiver's ciphertext and public key, because the noise flooding method guarantees sender privacy only when these materials are correctly generated. This validation can be performed using a zero-knowledge argument of knowledge

(ZKAoK) on the receiver's ciphertext and public key. In fact, a general compiler exists [AJLA+12] that transforms a semi-honestly secure BGV or BFV-based homomorphic evaluation protocol into a maliciously secure one, assuming the ideal functionality of a ZKAoK for BGV or BFV ciphertexts and public keys. Thus, improving the efficiency of ZKAoKs for the BGV and BFV schemes [HLSS25, CMS+23, BCOS20, AJLA+12] has been an important research topic, as the generation of ZKAoKs poses the primary overhead in achieving malicious security via the general compilation.

Challenges in CKKS. However, for the CKKS scheme, there have been no known formal methods for achieving sender privacy, and most applications claim sender privacy in a heuristic manner. In particular, there is no general randomization method for CKKS ciphertexts that is comparable to the noise flooding method used in exact HE schemes. Additionally, no ZKAoK has been proposed for CKKS ciphertexts and public keys, making the construction of maliciously secure CKKS-based protocols even more challenging.

The fundamental problem arises from the lack of a formally defined security goal that captures the approximate nature of CKKS. In exact HE-based protocols, the security goal is to construct an efficient simulator from the protocol output that is computationally indistinguishable from the adversary's view, which aligns with the standard security notion of multi-party computation (MPC) protocols. However, this security formulation does not apply to CKKS due to its unique decryption structure, where the plaintext and noise are fused [1], making it impossible for even the secret key owner to distinguish between them. As a result, the receiver always obtains a random approximation of the desired output, and constructing a simulator from this random approximation does not align with the standard MPC security notion.

Moreover, when considering malicious receivers, there is no known compilation method for CKKS-based protocols that enables sender privacy using ZKAoK functionality. Thus, we cannot precisely characterize the security enhancements that a ZKAoK for CKKS would provide against such adversaries. Additionally, even if such a compiler existed, no ZKAoK construction for CKKS ciphertexts and public keys has been proposed. Although ZKAoKs for BGV and BFV ciphertexts and public keys have been developed, they do not naturally extend to the CKKS case. For CKKS public keys, we need to prove that they are generated from sparse secret keys, which directly affects the precision of CKKS homomorphic operations and the performance of bootstrapping [CHK+18a], but there is no known method to handle this efficiently. Furthermore, for CKKS ciphertexts, the difficulty arises in proving the validity of the plaintexts used for encryption, which are typically expressed as constraints over complex numbers, while existing ZKAoKs for BGV or BFV only handle constraints over polynomial rings. Therefore, a fundamentally different approach is required to construct a ZKAoK for CKKS.

[1] Here, we do not consider discrete variants of CKKS, such as [BKSS24]

1.1 Our Contribution

In this paper, we address the challenges arising from CKKS-based secure function evaluation protocols. The key contributions of this work are as follows.

1. We define a new security notion called *differentially private homomorphic evaluation* (DPHE), which embraces the approximate nature of CKKS. In our definition, the security goal is relaxed to constructing a simulator from the protocol output whose max-divergence from the real protocol execution is bounded, mirroring the standard definition of differential privacy. The main implication of our definition is that if a protocol satisfies the DPHE property, then the evaluation results satisfy differential privacy with respect to the sender's private input. We observe that most applications of CKKS are PPML protocols, and the ultimate security goal of these PPML protocols is to achieve differential privacy. Thus, our relaxation suffices to provide the desired security guarantees in CKKS-based use cases. Additionally, our definition is formulated within the universal composability framework [Can01], and as a result, the security enhancements provided by ZKAoKs for CKKS against malicious receivers are naturally expressed in the form of a hybrid execution model.
2. We devise a general compilation method to achieve the new security notion based on the Laplace mechanism. To achieve this new security notion, one could employ the noise flooding method; however, in the CKKS context, this approach significantly degrades the usefulness of the protocol outputs, as it introduces exponentially large noise that corrupts the underlying plaintext. Instead, our compilation method leverages the Laplace mechanism, which introduces noise of moderate size drawn from a Laplace distribution. This allows the new security notion to be achieved without a significant degradation in the utility of the protocol output. Furthermore, our compilation method ensures sender privacy against malicious receivers in the hybrid execution model, which assumes the ideal functionality of a ZKAoK for CKKS.
3. We construct the first zero-knowledge argument of knowledge (ZKAoK) for CKKS in the form of a polynomial interactive oracle proof (PIOP) [CHM$^+$20] to handle malicious receivers. To validate the well-formedness of CKKS public keys, we design a new PIOP that efficiently verifies the sparsity of the corresponding secret keys. For CKKS ciphertexts, to prove the validity of the underlying plaintexts, we introduce a novel proof technique that incorporates homomorphic operations during verification. We also provide a concrete implementation and present benchmark results to demonstrate the practicality of our construction.

1.2 Related Work

DP-MPC. Similar to our definition of differentially private homomorphic evaluation, the concept of a differentially private multiparty computation (DP-MPC) protocol is discussed in [BNO08]. It defines the security of DP-MPC protocols

through the existence of an efficient simulator whose max-divergence is bounded with respect to the adversary's view in the semi-honest setting. However, its main focus is on how an existing secure MPC protocol can be transformed into a DP-MPC protocol by replacing the ideal functionality with a differentially private mechanism in the semi-honest setting. In contrast, our definition considers malicious receiver scenarios within the UC framework, and our DPHE compiler builds upon a CKKS homomorphic evaluation protocol, which is not inherently a secure MPC protocol. Thus, the concept of DP-MPC is not directly applicable to our setting.

Approximate MPC. Besides conventional secure multiparty computation, whose objective is to exactly evaluate a target function f, there is a generalization known as approximate MPC protocols [FIM+01], which considers scenarios where the protocol evaluates an approximation $\hat{f}$ instead of f. The motivation for this approach stems from situations where evaluating $\hat{f}$ can be done more efficiently within an MPC protocol, and the goal is to define a security notion ensuring that evaluating $\hat{f}$ does not reveal more information about the inputs than evaluating f. In this setting, the security of approximate MPC is defined to guarantee that the evaluation results $\hat{f}(x)$ can be derived solely from $f(x)$. However, in the case of CKKS-based evaluation protocols, the context is different because approximate evaluation is not a design choice but an inherent property of the CKKS scheme. Therefore, we consider the definition of approximate MPC to be incompatible with our setting.

ZKAoK for HE. As mentioned earlier, several works in the literature have constructed ZKAoKs for HE, especially for BGV and BFV. Boschini et al. [BCOS20] construct a proof system for BGV ciphertexts based on Aurora [BSCR+19]. Bell et al. [BGL+23] also construct a proof system for BGV ciphertexts based on the inner product argument of Bulletproofs [BBB+18]. Chatel et al. [CMS+23] construct a ZKAoK for BFV ciphertexts and public keys, based on the LANES [ALS20, ENS20, LNS20] framework, a lattice-based zero-knowledge proof system. The work most relevant to ours is [HLSS25], which constructs a PIOP-based ZKAoK for BFV ciphertexts and public keys. However, to the best of our knowledge, there has been no prior work proposing ZKAoKs for CKKS.

IND-CPA-D. Recently, a new security notion for CKKS, called IND-CPA-D, has been proposed [LM21]. Concrete attacks [GNSJ24, CCP+24, CSBB24] are continuously being raised, and preventive measures [LMSWS22] are being extensively studied. In short, the definition of IND-CPA-D primarily focuses on the case where decryption results of CKKS are shared with an evaluator, which may leak partial information about the secret key. In our case, we restrict our protocol to a sender-receiver model and assume that the sender does not receive any evaluation results, similar to other oblivious evaluation protocols [NP06, ADDG24]. Therefore, we do not consider IND-CPA-D security in our protocol, and we leave it as an orthogonal research direction.

Verifiable HE. Another research direction in HE-based secure function evaluation focuses on addressing the malicious behavior of the sender, often referred to as verifiable HE. One frequently used approach is the incorporation of suc-

cinct non-interactive arguments (SNARGs) to validate the sender's computation [BCFK21, FNP20, GNSV23, LATV12]. We believe that this approach can be naturally extended to our case, although in this paper, we focus on the semi-honest sender scenario.

2 Technical Overview

In this section, we provide a technical overview of our work. We begin by introducing our new security notion, differentially private homomorphic evaluation (DPHE), along with several of its implications. We then present a general compiler that transforms a plain CKKS-based protocol into a DPHE protocol. Finally, we outline the design of a ZKAoK for CKKS that validates the well-formedness of CKKS ciphertexts and public keys, which helps to achieve the DPHE property in the presence of malicious receivers.

2.1 Definition & Implication of DPHE

As discussed earlier, the most critical problem stems from the lack of a formally defined security goal tailored to CKKS. To address this, we introduce a new security notion, called a differentially private homomorphic evaluation (DPHE) protocol, which precisely captures the privacy requirements of CKKS-based applications.

Motivation. Our definition is motivated by the fact that most applications of CKKS are PPML inference protocols [LLL+22, JPK+24, BCCW19], which can be viewed as a privacy-enhanced variant of machine learning as a service (MLaaS) protocols, in which a receiver obtains inference results from a sender's model. In MLaaS protocols, the primary privacy concern stems from malicious receivers who may attempt to reconstruct the sender's training data or model parameters by issuing adversarial queries [CLE+19, SSSS17, CTW+21, CJM20]. To mitigate such attacks, privacy leakage is typically analyzed through the lens of differential privacy [ACG+16, LAG+19]. Based on this observation, we frame the ultimate security goal for sender privacy in CKKS-based protocols as ensuring differential privacy for the protocol output, particularly in PPML inference scenarios. To this end, we introduce a new security notion that formalizes this objective.

DP Prediction. Our starting point is the definition of differentially private prediction protocols from [DF18], which precisely captures the expected privacy guarantee for the sender in a MLaaS protocol in terms of differential privacy. To be precise, a DP prediction protocol is a MLaaS protocol in which the receiver has an input $x \in \mathcal{X}$, and the sender holds a private input $\theta \in \Theta$, such as model parameters or training data used to construct the model. An algorithm $\mathtt{M}_\theta : \mathcal{X} \to \mathcal{Y}$, parametrized by θ, is said to be ε-differentially private for prediction if the following condition holds for all adjacent pairs $\theta, \theta' \in \Theta$, inputs $x \in \mathcal{X}$, probabilistic polynomial-time (PPT) algorithms $\mathtt{D}$, and security parameter λ.

$$\Pr[\mathtt{D}(\mathtt{M}_\theta(x)) = 1] \leq e^\varepsilon \cdot \Pr[\mathtt{D}(\mathtt{M}_{\theta'}(x)) = 1] + \mathsf{negl}(\lambda)$$

We note that we reformulate the original definition in the context of computational differential privacy [MPRV09] to make it applicable within cryptographic protocols.

Definition of DPHE. Based on DP prediction protocols, we extend the scenario to protect the privacy of the receiver's input. To be precise, the receiver's input x is provided as a ciphertext $\mathtt{Enc}(x)$ via CKKS encryption, and the evaluation of $\mathtt{M}_\theta$ is performed using the homomorphic evaluation algorithm $\mathtt{Eval}(\mathtt{M}_\theta, \cdot)$. In this setting, our goal is to characterize the conditions on $\mathtt{Eval}$ under which the DP prediction property is preserved. To this end, we model the ideal functionality $\mathtt{F}_\theta$ as a sender-receiver protocol that takes input x from the receiver and outputs $\mathtt{M}_\theta(x)$ to the receiver. The real-world execution is defined as a protocol Π_θ, where the sender receives $\mathtt{Enc}(x)$ from the receiver and returns $\mathtt{Eval}(\mathtt{M}_\theta, \mathtt{Enc}(x))$ to the receiver. Then, Π_θ is said to be a *ε'-differentially private homomorphic evaluation* (DPHE) protocol if there exists a PPT simulator $\mathtt{S}$ such that, for all PPT adversaries $\mathtt{A}$ and PPT environments $\mathtt{Z}$, the following conditions are satisfied for all PPT algorithms $\mathtt{D}$.

$$\Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_\theta, \mathtt{A}, \mathtt{Z}]) = 1\right] \leq e^{\varepsilon'} \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}, \mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda)$$

$$\Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}, \mathtt{Z}]) = 1\right] \leq e^{\varepsilon'} \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_\theta, \mathtt{A}, \mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda)$$

These conditions essentially state the existence of an efficient simulator $\mathtt{S}$ such that the max-divergence between the execution of the ideal functionality and the real protocol is bounded by ε'.

Implication of DPHE. The definition of DPHE implies the following two key properties.

1. If $\mathtt{M}_\theta$ is a ε-DP prediction protocol and Π_θ is a ε'-DPHE protocol, then Π_θ is a $(\varepsilon + 2\varepsilon')$-DP prediction protocol for all PPT receivers.
2. Let Π be a ε'-DPHE protocol in the $\mathtt{F}$-hybrid model. Then, the protocol Π', which replaces $\mathtt{F}$ with its secure implementation $\Pi_\mathtt{F}$, is also a ε'-DPHE protocol.

The first property ensures that the DPHE definition provides the intended security guarantees for a DPHE protocol, namely, that it preserves the DP prediction property. The second property demonstrates that the DPHE definition supports subprotocol composition, as in the UC framework [Can01], which enables the integration of ZKAoK functionality in the design of DPHE protocols.

2.2 DPHE Compiler Based on the Laplace Mechanism

To instantiate a DPHE protocol, we develop an efficient compiler that transforms a plain CKKS-based homomorphic evaluation protocol into a DPHE protocol. Our compiler is based on the Laplace mechanism and operates within the hybrid execution model that assumes the ideal functionality of a ZKAoK for CKKS. Let $f_\theta : \mathcal{X} \to \mathcal{Y}$ be a function parameterized by $\theta \in \Theta$ that has bounded sensitivity

with respect to θ, i.e., there exists a fixed $\xi > 0$ such that $|f_\theta(x) - f_{\theta'}(x)| \leq \xi$ for all adjacent $\theta, \theta' \in \Theta$ and all inputs $x \in \mathcal{X}$. Consider a plain CKKS-based protocol that evaluates the function f_θ using the CKKS homomorphic evaluation algorithm $\texttt{Eval}$. We can then compile it into a DPHE protocol by having the sender run the following algorithm, where t is a noise value drawn from a Laplace distribution, $\Delta > 0$ is an integer used as a scaling factor, and $\texttt{Enc}(0)$ denotes a fresh encryption of zero.

$$\texttt{Eval}'(f_\theta, \texttt{Enc}(x)) := \Delta \cdot \texttt{Eval}(f_\theta + t, \texttt{Enc}(x)) + \texttt{Enc}(0)$$

Core Idea. The core idea of the above algorithm is as follows. Let $g_\theta(x) := \texttt{Dec}(\texttt{Eval}(f_\theta, \texttt{Enc}(x)))$, where $\texttt{Dec}$ denotes the decryption algorithm. Then, it holds that $g_\theta(x) = f_\theta(x) + e$ for some approximation noise e. Assuming that the input ciphertext $\texttt{Enc}(x)$ and the public keys used in $\texttt{Eval}$ are well-formed, we can estimate a bound B on the noise e. In this case, g_θ becomes a function with bounded sensitivity $\xi + B$, and we can apply the Laplace mechanism to it by adding noise t drawn from a Laplace distribution with parameter $(\xi + B)/\varepsilon$, where ε is the target differential privacy (DP) parameter. However, the Laplace mechanism alone is not sufficient to achieve the DPHE property, since the receiver obtains a ciphertext rather than the decrypted result in the protocol. To randomize the ciphertext components, we need to add an encryption of zero, but this introduces additional noise in the case of CKKS ciphertexts. Therefore, we multiply the scaling factor Δ before adding the encryption of zero, in order to decouple the evaluation result from the encryption noise. As a result, we construct a DPHE compiler based on the Laplace mechanism.

Role of ZKAoK. We note that our DPHE compiler guarantees sender privacy only when the CKKS ciphertexts and public keys are well-formed. Therefore, to defend against a malicious receiver who may submit ill-formed inputs, we require a ZKAoK functionality $\texttt{F}_{\texttt{ZKAoK}}$ that validates the well-formedness of CKKS ciphertexts and public keys. In the $\texttt{F}_{\texttt{ZKAoK}}$-hybrid model, our DPHE compiler ensures sender privacy against malicious receivers, as the sender can be assured that the receiver's inputs are well-formed. This precisely describes the role of ZKAoK for CKKS in enabling the DPHE property in the presence of malicious adversaries.

2.3 Efficient ZKAoK for CKKS

To concretely instantiate our DPHE compiler, a concrete implementation of a ZKAoK for CKKS is required to replace the assumption of ideal functionality. Therefore, we construct a ZKAoK for CKKS to address this remaining challenge. We emphasize that our construction is the first to prove the well-formedness of CKKS public keys and ciphertexts.

ZKAoK for BFV. Our starting point is the previous construction of a ZKAoK for BFV in [HLSS25]. To prove the well-formedness of BFV public keys and ciphertexts, it designs a ZKAoK that proves the arithmetic constraints defined

over the polynomial ring $R_q = \mathbb{Z}_q[X]/(X^N + 1)$, as well as the L^∞-norm bound of the witness polynomials in R_q. It particularly leverages the polynomial interactive oracle proof (PIOP) framework from [CHM+20, BSCR+19] to take advantage of its expressiveness in the design of the ZKAoK. However, the previous construction has limitations when extended to a ZKAoK for CKKS public keys and ciphertexts. For CKKS public keys, it does not provide tools for proving the sparsity of the underlying secret key, which is necessary for ensuring the well-formedness of CKKS public keys. For CKKS ciphertexts, it lacks a proof technique for handling constraints on the underlying plaintexts, which is essential for verifying the well-formedness of CKKS ciphertexts.

ZKAoK for CKKS Public Keys. To address the issues related to CKKS public keys, we devise a new PIOP that can efficiently verify the sparsity of the secret key. In CKKS, a sparse ternary secret key $\boldsymbol{s} \in R_q$ is typically used, where each coefficient is either 0 or ± 1, and the number of non-zero coefficients is bounded by h, i.e., $\|\boldsymbol{s}\|_\infty \leq 1$ and $\|\boldsymbol{s}\|_1 \leq h$ hold. Although the first condition can be verified using the PIOP introduced in [HLSS25], there is no known PIOP that can efficiently check the latter. To address this, we observe that $\|\boldsymbol{s}\|_1 \leq h \iff \|\boldsymbol{s}\|_2^2 \leq h$ when $\|\boldsymbol{s}\|_\infty = 1$, and bounds on the L^2-norm can be efficiently verified by adapting the univariate sumcheck PIOP from [BSCR+19]. As a result, we construct a new PIOP that can efficiently verify the sparsity of secret keys when validating CKKS public keys.

ZKAoK for CKKS Ciphertexts. In the case of a ZKAoK for CKKS ciphertexts, the main difficulty arises from proving the validity of the plaintext used for encryption. To be precise, a message is a vector $\vec{m} \in \mathbb{C}^{N/2}$, and the packing method $\mathtt{Pack}$ transforms it into a plaintext $\boldsymbol{m} = \mathtt{Pack}(\vec{m})$, which resides in the polynomial ring R_q. In the context of our DPHE compiler, to ensure the well-formedness of the ciphertext, we need to prove that the plaintext $\boldsymbol{m}$ used for encryption unpacks to $\vec{m} = \mathtt{Unpack}(\boldsymbol{m}) \in \mathcal{X}$, where $\mathcal{X} \subseteq \mathbb{C}^{N/2}$ is the input domain, typically represented as closed cubes in $\mathbb{R}^{N/2}$.

To resolve this issue, we devise a new proof technique that employs the homomorphic operation $\mathtt{CtoS}$, known as coeff-to-slot, during the verification process. Although this operation was initially proposed for CKKS bootstrapping [CHK+18a], we repurpose it to enable an efficient ZKAoK construction. In short, the coeff-to-slot operation takes one input ciphertext ct, whose plaintext is $\boldsymbol{m} = \sum_{i=0}^{N-1} m_i X^i \in R_q$, and outputs two ciphertexts ct' and ct'' whose plaintexts are $\boldsymbol{m}'$ and $\boldsymbol{m}''$, satisfying $\mathtt{Unpack}(\boldsymbol{m}') \approx \frac{1}{\Delta}(m_0, \ldots, m_{N/2-1})$ and $\mathtt{Unpack}(\boldsymbol{m}'') \approx \frac{1}{\Delta}(m_{N/2}, \ldots, m_{N-1})$ for some scaling factor $\Delta > 0$. The core idea is that if we can prove that the coefficient vector $(m_0, \ldots, m_{N-1})$ lies in the scaled domain $\lfloor \Delta \cdot \mathcal{X} \rceil^2$, then we can ensure that $\mathtt{Unpack}(\boldsymbol{m}')$ and $\mathtt{Unpack}(\boldsymbol{m}'')$ lie in the input domain $\mathcal{X}$. Verifying that the coefficients of the plaintext lie within a certain range can be accomplished using the PIOP from [HLSS25], which checks the L^∞-norm. In other words, if a sender generates a ciphertext ct that encrypts $\boldsymbol{m}$ and proves that the coefficient vector of $\boldsymbol{m}$ lies in $\lfloor \Delta \cdot \mathcal{X} \rceil^2$, then the receiver can obtain well-formed ciphertexts ct' and ct'' via the coeff-to-slot operation, whose plaintexts lie in $\mathcal{X}$. As a result, we can instantiate a ZKAoK

functionality for CKKS ciphertexts by utilizing the coeff-to-slot operation in the verification process.

Implementation. To demonstrate the practicality of our ZKAoK for CKKS, we present a proof-of-concept implementation. Our construction is not only the first to design an efficient ZKAoK for CKKS but also the first to provide a concrete instantiation of it. For the implementation, we utilize the lattice-based polynomial commitment scheme (PCS) [HSS24] to compile PIOPs, as it supports homomorphic encryption–friendly parameters. The benchmark results show that our implementation achieves a proof size of approximately half the combined size of the ciphertext and public key, and that proof generation and verification can be completed in just a few minutes.

3 Preliminaries

3.1 Notation

For a positive integer q, we use $\mathbb{Z} \cap (-q/2, q/2]$ as a representative set of $\mathbb{Z}_q$, and denote by $[a]_q$ the reduction of a modulo q. Vectors over $\mathbb{Z}$ or $\mathbb{Z}_q$ are denoted with regular lowercase letters and arrows, such as $\vec{v}$, and matrices over $\mathbb{Z}$ or $\mathbb{Z}_q$ are represented by regular uppercase letters. We regard all vectors as column vectors, and we use the symbol $\|$ for the concatenation of two vectors.

Let N be a power of two. We denote by $R = \mathbb{Z}[X]/(X^N + 1)$ the ring of integers of the $2N$-th cyclotomic field, and by $R_q = R/qR$ the residue ring of R modulo q. For polynomials, we use bold lowercase letters to denote them e.g., $\boldsymbol{f}$. For a vector $\vec{v} = (v_0, \ldots, v_{n-1}) \in \mathbb{Z}^n$, the L^p and L^∞ norms are defined as $\|\vec{v}\|_p := \sqrt[p]{\sum_{i=0}^{n-1} |v_i|^p}$ and $\|\vec{v}\|_\infty := \max_{0 \le i < n} |v_i|$ for $p \ge 1$. The Hadamard product is denoted by $\odot$. For a polynomial $\boldsymbol{f}$ or a vector of polynomials $\vec{\boldsymbol{f}}$, $\|\boldsymbol{f}\|_p$ and $\left\|\vec{\boldsymbol{f}}\right\|_p$ are calculated by regarding them as coefficient vectors. For a matrix $A \in \mathbb{R}^{n \times n}$, we denote the matrix norm of A by $\|A\|_2 := \max_{0 \neq \vec{x} \in \mathbb{R}^n} \frac{\|A\vec{x}\|_2}{\|\vec{x}\|_2}$.

3.2 Probability Distributions

We denote sampling x from the distribution $\mathcal{D}$ by $x \leftarrow \mathcal{D}$. For distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over a countable set S (e.g. $\mathbb{Z}^n$), the statistical distance of $\mathcal{D}_1$ and $\mathcal{D}_2$ is defined as $\frac{1}{2} \cdot \sum_{x \in S} |\mathcal{D}_1(x) - \mathcal{D}_2(x)| \in [0, 1]$. We denote the uniform distribution over S by $\mathcal{U}(S)$ when S is finite.

For $\sigma > 0$, we define the Gaussian function $\rho_\sigma : \mathbb{R} \to (0, 1]$ as $\rho_\sigma(x) := \exp(-\pi \cdot (x/\sigma)^2)$. The discrete Gaussian distribution $\mathtt{DG}(\sigma)$ with parameter σ is defined as a distribution over $\mathbb{Z}$, whose probability mass function is $\rho_\sigma(x)/\rho_\sigma(\mathbb{Z})$, where $\rho_\sigma(\mathbb{Z}) := \sum_{x \in \mathbb{Z}} \rho_\sigma(x) < \infty$. For a polynomial $\boldsymbol{f}$ of degree d, we denote by $\boldsymbol{f} \leftarrow \mathtt{DG}(\sigma)^d$ if each of its coefficients is sampled from $\mathtt{DG}(\sigma)$. For $b > 0$, the Laplace distribution $\mathtt{Lap}(b)$ with parameter b is defined as a distribution over $\mathbb{R}$, whose probability density function is $\frac{1}{2b} \exp(-|x|/b)$.

3.3 The RLWE Problem

We review the definition of the RLWE problem [LPR10], as well as its variant, the Hint-RLWE problem [KLSS23]. Under appropriate parameter settings, the Hint-RLWE problem can be reduced from the RLWE problem.

Definition 1 ([LPR10, RLWE]). *Let χ_s, χ_e be distributions over R. Then, the goal of the Ring-LWE (MLWE) problem is to distinguish $(\boldsymbol{a}, \boldsymbol{u})$ from $(\boldsymbol{a}, \boldsymbol{as}+\boldsymbol{e})$ for $\boldsymbol{a} \leftarrow \mathcal{U}(R_q)$, $\boldsymbol{e} \leftarrow \chi_e$, and $\boldsymbol{s} \leftarrow \chi_s$. We say that a PPT algorithm $\mathtt{D}$ has an advantage ε in solving $\mathsf{RLWE}_{q,\chi_s,\chi_e}$ if the following holds.*

$$\left| \Pr[\mathtt{D}(\boldsymbol{a}, \boldsymbol{u}) = 1] - \Pr[\mathtt{D}(\boldsymbol{a}, \boldsymbol{as}+\boldsymbol{e}) = 1] \right| \geq \varepsilon$$

When χ_s, χ_e are discrete Gaussian distributions, we denote them simply by their width parameters.

Definition 2 ([KLSS23, Hint-RLWE]). *Let $\boldsymbol{c}_s, \boldsymbol{c}_e$ be elements in R, and let $\chi_s, \chi_e, \chi_f, \chi_g$ be distributions over R. Then, the goal of the Hint-RLWE problem is to distinguish $(\boldsymbol{a},\ \boldsymbol{u},\ \boldsymbol{c}_s\boldsymbol{s}+\boldsymbol{f},\ \boldsymbol{c}_e\boldsymbol{e}+\boldsymbol{g})$ from $(\boldsymbol{a},\ \boldsymbol{as}+\boldsymbol{e},\ \boldsymbol{c}_s\boldsymbol{s}+\boldsymbol{f},\ \boldsymbol{c}_e\boldsymbol{e}+\boldsymbol{g})$ for $\boldsymbol{a} \leftarrow \mathcal{U}(R_q)$, $\boldsymbol{e} \leftarrow \chi_e$, $\boldsymbol{f} \leftarrow \chi_f$, $\boldsymbol{g} \leftarrow \chi_g$ and $\boldsymbol{s} \leftarrow \chi_s$. We say that a PPT algorithm $\mathtt{D}$ has an advantage ε in solving $\mathsf{HintRLWE}^{\boldsymbol{c}_s,\boldsymbol{c}_e,\chi_f,\chi_g}_{q,\chi_s,\chi_e}$ if the following holds.*

$$\left| \Pr[\mathtt{D}(\boldsymbol{a},\ \boldsymbol{u},\ \boldsymbol{c}_s\boldsymbol{s}+\boldsymbol{f},\ \boldsymbol{c}_e\boldsymbol{e}+\boldsymbol{g}) = 1] - \Pr[\mathtt{D}(\boldsymbol{a},\ \boldsymbol{as}+\boldsymbol{e},\ \boldsymbol{c}_s\boldsymbol{s}+\boldsymbol{f},\ \boldsymbol{c}_e\boldsymbol{e}+\boldsymbol{g}) = 1] \right| \geq \varepsilon$$

Definition 3 ([MR07, Smoothing Parameter]). *For an n-dimensional lattice Λ and $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(\Lambda)$ is the smallest s such that $\rho_{1/s}(\Lambda^* \backslash \{\vec{0}\}) \leq \varepsilon$.*

Theorem 1 ([KLSS23, Theorem 1]). *Let $\sigma_e, \sigma_s > 0$, and $\sigma_f, \sigma_g, \sigma > 0$ be real numbers, which satisfy the following, where $\|\boldsymbol{c}_s\|_1 \leq B_s$ and $\|\boldsymbol{c}_e\|_1 \leq B_e$.*

$$\frac{1}{\sigma^2} = 2\left(\frac{1}{\sigma_s^2} + \frac{B_s^2}{\sigma_f^2}\right) = 2\left(\frac{1}{\sigma_e^2} + \frac{B_e^2}{\sigma_g^2}\right)$$

If $\sigma \geq \sqrt{2} \cdot \eta_{(1/2^\lambda)}(\mathbb{Z}^N)$, then there exists an efficient reduction from $\mathsf{RLWE}_{q,\sigma,\sigma}$ to $\mathsf{HintRLWE}^{\boldsymbol{c}_s,\boldsymbol{c}_e,\sigma_f,\sigma_g}_{q,\sigma_s,\sigma_e}$ that reduces the advantage by at most $\mathsf{negl}(\lambda)$.

3.4 Differential Privacy

Below, we review the definition of differential privacy (DP) [DMNS06] and its extension to computationally bounded adversaries, known as computational differential privacy (CDP) [MPRV09].

Definition 4 ([DMNS06, Differential Privacy]). *A randomized algorithm $\mathtt{f} : \mathcal{X} \to \mathcal{Y}$ provides ϵ-differential privacy (DP) if for all adjacent inputs $x, x' \in \mathcal{X}$, and all subsets $S \subseteq \mathcal{Y}$, the following holds.*

$$\Pr[\mathtt{f}(x) \in S] \leq e^\epsilon \cdot \Pr[\mathtt{f}(x') \in S].$$

Definition 5 ([MPRV09, Computational Differential Privacy]). *A randomized algorithm* $\mathtt{f} : \mathcal{X} \to \mathcal{Y}$ *provides* ε*-computational differential privacy (CDP) if for all adjacent inputs* $x, x' \in \mathcal{X}$*, the following holds for all PPT algorithms* $\mathtt{D}$*.*

$$\Pr[\mathtt{D}(\mathtt{f}(x)) = 1] \leq e^{\varepsilon} \cdot \Pr[\mathtt{D}(\mathtt{f}(x')) = 1] + \mathsf{negl}(\lambda)$$

3.5 The CKKS Scheme

We review the CKKS scheme [CKKS17], especially for its residue number system (RNS)-variant [CHK+18b].

Basic Operations. The CKKS scheme is an encryption scheme whose plaintext space is R, and ciphertext space is $R_{Q_L}^2$. Basic operations, such as setup, key generation, encryption, and decryption algorithms, are defined as follows.

- $\underline{\mathtt{Setup}}(1^\lambda, N) \to \mathsf{pp}$: Given a security parameter λ and a ring dimension N, choose an RNS modulus chain $q_0, q_1, \ldots, q_L$, a secret key distribution χ_s, an error distribution χ_e, and a scaling factor $\Delta \in \mathbb{Z}$. Output a public parameter $\mathsf{pp} = (N, \Delta, Q_L = \prod_{i=0}^{L} q_i, \chi_s, \chi_e)$.
- $\underline{\mathtt{KeyGen}}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{pk})$: Given a public parameter pp and a set of automorphisms, generate a secret key sk and a public key pk as follows, where $\mathsf{pk} = (\mathsf{ek}, \mathsf{rlk}, \mathsf{rtk}, \mathsf{cjk})$, and $\vec{g} = (Q_L/q_0, \ldots, Q_L/q_{L-1}) \in \mathbb{Z}^L$.
  1. Sample $\boldsymbol{s} \leftarrow \chi_s$, and set a secret key $\mathsf{sk} = \boldsymbol{s} \in R$.
  2. Sample $\boldsymbol{u}_{\mathsf{ek}} \leftarrow \mathcal{U}(R_{Q_L})$, $\boldsymbol{e}_{\mathsf{ek}} \leftarrow \chi_e$, and set an encryption key $\mathsf{ek} = (-\boldsymbol{u}_{\mathsf{ek}}\boldsymbol{s} + \boldsymbol{e}_{\mathsf{ek}},\ \boldsymbol{u}_{\mathsf{ek}}) \in R_{Q_L}^2$.
  3. Sample $\vec{\boldsymbol{u}}_{\mathsf{rlk}} \leftarrow \mathcal{U}(R_{Q_L}^L)$, $\vec{\boldsymbol{e}}_{\mathsf{rlk}} \leftarrow \chi_e^L$, and set a relinearization key $\mathsf{rlk} = (-\boldsymbol{s} \cdot \boldsymbol{u}_{\mathsf{rlk}} + \boldsymbol{s}^2 \cdot \vec{g} + \vec{\boldsymbol{e}}_{\mathsf{rlk}},\ \vec{\boldsymbol{u}}_{\mathsf{rlk}}) \in R_{Q_L}^{2L}$.
  4. Sample $\vec{\boldsymbol{u}}_{\mathsf{rtk}} \leftarrow \mathcal{U}(R_{Q_L}^L)$, $\vec{\boldsymbol{e}}_{\mathsf{rtk}} \leftarrow \chi_e^L$, and set a rotation key $\mathsf{rtk} = (-\boldsymbol{s} \cdot \boldsymbol{u}_{\mathsf{rtk}} + \varphi(\boldsymbol{s}) \cdot \vec{g} + \vec{\boldsymbol{e}}_{\mathsf{rtk}},\ \vec{\boldsymbol{u}}_{\mathsf{rtk}}) \in R_{Q_L}^{2L}$, where $\varphi : X \mapsto X^5$ is an automorphism over R.
  5. Sample $\vec{\boldsymbol{u}}_{\mathsf{cjk}} \leftarrow \mathcal{U}(R_{Q_L}^L)$, $\vec{\boldsymbol{e}}_{\mathsf{cjk}} \leftarrow \chi_e^L$, and set a conjugation key $\mathsf{cjk} = (-\boldsymbol{s} \cdot \boldsymbol{u}_{\mathsf{cjk}} + \psi(\boldsymbol{s}) \cdot \vec{g} + \vec{\boldsymbol{e}}_{\mathsf{cjk}},\ \vec{\boldsymbol{u}}_{\mathsf{cjk}}) \in R_{Q_L}^{2L}$, where $\psi : X \mapsto X^{-1}$ is an automorphism over R.
- $\underline{\mathtt{Enc}_{\mathsf{sk}}}(\boldsymbol{m}) \to \mathsf{ct}$: Given a secret key $\mathsf{sk} = \boldsymbol{s}$ and a plaintext $\boldsymbol{m} \in R$, output a ciphertext $\mathsf{ct} = (-\boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{m} + \boldsymbol{e}, \boldsymbol{a}) \in R_{Q_L}^2$, where $\boldsymbol{a} \leftarrow \mathcal{U}(R_{Q_L})$, and $\boldsymbol{e} \leftarrow \chi_e$.
- $\underline{\mathtt{Dec}_{\mathsf{sk}}}(\mathsf{ct}) \to \boldsymbol{m}$: Given a secret key $\mathsf{sk} = \boldsymbol{s}$ and a ciphertext $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in R_{Q_\ell}^2$, output a plaintext $\boldsymbol{m} = \boldsymbol{c}_0 + \boldsymbol{c}_1\boldsymbol{s} \pmod{Q_0}$.

We note that the CKKS scheme achieves IND-CPA security under the computational hardness of $\mathsf{RLWE}_{Q_L, \chi_s, \chi_e}$, assuming circular security [MV24]. In practice, we usually use a sparse ternary distribution $\mathsf{HWT}(h)$ for a secret key distribution χ_s, where each coefficient is in $\{-1, 0, 1\}$ and the Hamming weight is bounded by h, since it provides smaller noise growth after each homomorphic operation and efficient bootstrapping performance [CHK+18a].

Homomorphic Operations. The CKKS scheme supports homomorphic addition, multiplication, automorphism, and rounding operations over R in an approximate manner. For automorphism operations, there are two types: rotation and conjugation, which evaluate the automorphisms $\varphi : X \mapsto X^5$ and $\psi : X \mapsto X^{-1}$, respectively. In the CKKS scheme, one can emulate arithmetic over $\mathbb{R}[X]/(X^N+1)$ in fixed-point arithmetic over $R = \mathbb{Z}[X]/(X^N+1)$ thanks to the homomorphic rounding operations, which is a distinctive feature compared to other HE schemes [BGV12, Bra12, FV12, CGGI20].

Definition 6 (External Product). *Let $Q_\ell = \prod_{i=0}^{\ell} q_i$ for $0 \le \ell < L$. An external product is a binary operation $\boxdot : R_{Q_\ell} \times R_{Q_L}^L \to R_{Q_\ell}$, defined as follows for $\boldsymbol{a} \in R_{Q_\ell}$ and $\vec{\boldsymbol{u}} = (\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{L-1}) \in R_{Q_L}^L$.*

$$\boldsymbol{a} \boxdot \vec{\boldsymbol{u}} = \left\lfloor \frac{1}{q_L} \Big(\sum_{i=0}^{\ell-1} [(Q_{L-1}/q_i)^{-1} \cdot \boldsymbol{a}]_{q_i} \cdot \boldsymbol{u}_i \Big) \right\rceil \pmod{Q_\ell}$$

By an abuse of notation, we write $\boldsymbol{a} \boxdot (\vec{\boldsymbol{u}}_0, \vec{\boldsymbol{u}}_1) = (\boldsymbol{a} \boxdot \vec{\boldsymbol{u}}_0, \boldsymbol{a} \boxdot \vec{\boldsymbol{u}}_1)$ for $\vec{\boldsymbol{u}}_0, \vec{\boldsymbol{u}}_1 \in R_{Q_L}^L$.

- $\underline{\mathtt{Add}}(\mathsf{ct}, \mathsf{ct}') \to \mathsf{ct}_{add}$: Given ciphertexts $\mathsf{ct}, \mathsf{ct}' \in R_{Q_\ell}^2$, output a ciphertext $\mathsf{ct}_{add} = \mathsf{ct} + \mathsf{ct}' \in R_{Q_\ell}^2$.
- $\underline{\mathtt{Mul}_{\mathsf{rlk}}}(\mathsf{ct}, \mathsf{ct}') \to \mathsf{ct}_{mul}$: Given ciphertexts $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1), \mathsf{ct}' = (\boldsymbol{c}_0', \boldsymbol{c}_1') \in R_{Q_\ell}^2$ for $\ell < L$, let $\boldsymbol{d}_0 = \boldsymbol{c}_0\boldsymbol{c}_0'$, $\boldsymbol{d}_1 = \boldsymbol{c}_0\boldsymbol{c}_1' + \boldsymbol{c}_0'\boldsymbol{c}_1$, and $\boldsymbol{d}_2 = \boldsymbol{c}_1\boldsymbol{c}_1'$. Output a ciphertext $\mathsf{ct}_{mul} = (\boldsymbol{d}_0, \boldsymbol{d}_1) + \boldsymbol{d}_2 \boxdot \mathsf{rlk} \in R_{Q_\ell}^2$
- $\underline{\mathtt{Rot}_{\mathsf{rtk}}}(\mathsf{ct}) \to \mathsf{ct}'$: Given ciphertexts $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in R_{Q_\ell}^2$ for $\ell < L$, output a ciphertext $\mathsf{ct}' = (\varphi(\boldsymbol{c}_0), 0) + \varphi(\boldsymbol{c}_1) \boxdot \mathsf{rtk} \in R_{Q_\ell}^2$.
- $\underline{\mathtt{Conj}_{\mathsf{cjk}}}(\mathsf{ct}) \to \mathsf{ct}'$: Given ciphertexts $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in R_{Q_\ell}^2$ for $\ell < L$, output a ciphertext $\mathsf{ct}' = (\psi(\boldsymbol{c}_0), 0) + \psi(\boldsymbol{c}_1) \boxdot \mathsf{cjk} \in R_{Q_\ell}^2$.
- $\underline{\mathtt{Round}}(\mathsf{ct}) \to \mathsf{ct}'$: Given a ciphertext $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in R_{Q_\ell}^2$ for $\ell \ge 1$, output a ciphertext $\mathsf{ct}' = (\lfloor \boldsymbol{c}_0/q_\ell \rceil, \lfloor \boldsymbol{c}_1/q_\ell \rceil) \in R_{Q_{\ell-1}}^2$.

Since homomorphic operations in CKKS work in an approximate manner, small errors are introduced after each homomorphic operation. To quantify these errors, we use the following notation.

Definition 7 (Ciphertext Error). *For a CKKS ciphertext $\mathsf{ct} \in R_{Q_\ell}^2$, a plaintext $\boldsymbol{m} \in R$, and a secret key sk, the ciphertext error of $(\mathsf{ct}, \boldsymbol{m}, \mathsf{sk})$ is defined as follows.*

$$\mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}, \boldsymbol{m}) = \|\mathtt{Dec}_{\mathsf{sk}}(\mathsf{ct}) - \boldsymbol{m}\|_\infty$$

In Appendix A, we provide a detailed analysis of error growth for each homomorphic operation. These analyses imply that, given upper bounds on the Hamming weight of the secret key sk, the errors used in generating the public key pk, and the errors and plaintexts of the input ciphertexts, we can estimate an error bound for the final output ciphertext when homomorphically evaluating a complex circuit in CKKS. We formalize this as follows.

Theorem 2 (Error Bound Estimation). *Let* $\mathtt{C} : R^k \to R$ *be a* k*-ary admissible circuit for the CKKS scheme, i.e., a circuit that can be evaluated using addition, multiplication, automorphism, and rounding operations over* R*. Let* $\mathsf{ct}_0, \ldots, \mathsf{ct}_{k-1}$ *and* $\boldsymbol{m}_0, \ldots, \boldsymbol{m}_{k-1}$ *be such that* $\mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}_i, \boldsymbol{m}_i) \leq B_e$*, and* $\|\boldsymbol{m}_i\|_\infty \leq B_m$*. Furthermore, let* $\|\vec{\boldsymbol{e}}_{\mathsf{rlk}}\|_\infty \leq B_{\mathsf{rlk}}$*,* $\|\vec{\boldsymbol{e}}_{\mathsf{rtk}}\|_\infty \leq B_{\mathsf{rtk}}$*,* $\|\vec{\boldsymbol{e}}_{\mathsf{cjk}}\|_\infty \leq B_{\mathsf{cjk}}$*, and* $\|\boldsymbol{s}\|_\infty \leq h$ *for a public key* $\mathsf{pk} = (\mathsf{ek}, \mathsf{rlk}, \mathsf{rtk}, \mathsf{cjk})$ *and a secret key* $\mathsf{sk} = \boldsymbol{s}$*. Then, there exists an algorithm* $\mathtt{Estim}$ *such that the following holds.*

$$\mathtt{Err}\Big(\mathtt{Eval}_{\mathsf{pk}}(\mathtt{C}; \mathsf{ct}_0, \cdots, \mathsf{ct}_{k-1}), \mathtt{C}(\boldsymbol{m}_0, \ldots, \boldsymbol{m}_{k-1})\Big) \leq \mathtt{Estim}(\mathtt{C}; h, B_{\mathsf{rlk}}, B_{\mathsf{rtk}}, B_{\mathsf{cjk}}, B_e, B_m)$$

Proof. We defer the proof to Appendix A.

Packing Method. The CKKS scheme supports the message space $\mathbb{C}^{N/2}$ through an isomorphism $\iota : \mathbb{R}[X]/(X^N + 1) \to \mathbb{C}^{N/2}$, called the canonical embedding. Combined with fixed-point emulation of $\mathbb{R}[X]/(X^N+1)$ in R, the CKKS scheme supports SIMD operations in the message space $\mathbb{C}^{N/2}$ in an approximate manner. The following algorithms describe how to convert a message in $\mathbb{C}^{N/2}$ to a plaintext in $\mathbb{R}$ and vice versa.

- $\underline{\mathtt{Pack}}(\vec{m}) \to \boldsymbol{m}$: Given a message vector $\vec{m} \in \mathbb{C}^{N/2}$, output a plaintext $\boldsymbol{m} = \lfloor \Delta \cdot \iota^{-1}(\vec{m}) \rceil \in R$
- $\overline{\mathtt{Unpack}}(\boldsymbol{m}) \to \vec{m}$: Given a plaintext $\boldsymbol{m} \in R$, output a message vector $\vec{m} = \frac{1}{\Delta} \cdot \iota(\boldsymbol{m}) \in \mathbb{C}^{N/2}$

3.6 Interactive Argument of Knowledge

We use the standard definition of an interactive argument of knowledge with the honest-verifier zero-knowledge (HVZK) property. The full definition can be found in Appendix B. Next, we review the definition of a polynomial interactive oracle proof (PIOP) from [CHM+20], which is a special class of interactive oracle proof [BSCS16]. In the following definition, we restrict the polynomials to be univariate.

Definition 8 (Polynomial Interactive Oracle Proof). *Let* $\Pi = (\mathtt{Setup}, \mathtt{P}, \mathtt{V})$ *be a public-coin interactive oracle proof system for a relation* R*, and let* $\mathbb{F}$ *be a finite field.* Π *is called a polynomial interactive oracle proof (PIOP), which satisfies the followings.*

- *Every message from the prover is a polynomial oracle* $([\![\boldsymbol{f}]\!], d)$*, where* $\boldsymbol{f} \in \mathbb{F}[X]$ *of degree* $\leq d$*.*
- *Every message from the verifier is a random challenge.*
- *The verifier has oracle access to evaluations of the prover's polynomials at arbitrary points.*

Π *is called a honest verifier zero-knowledge PIOP if it is an HVZK argument of knowledge, where* $\mathtt{View}$ *outputs the messages from the verifier and the responses to polynomial evaluation queries.*

There exists a general compilation method [CHM+20] that transforms a PIOP into an interactive argument of knowledge, with the aid of a polynomial commitment scheme. We defer the details of the PIOP compilation to Appendix B.

4 PIOP Toolbox

In this section, we present a PIOP toolbox for designing ZKAoKs for CKKS. We begin by reviewing the prior work [HLSS25], which constructs a ZKAoK for BFV based on the PIOP framework. We then introduce a novel PIOP for proving L^2-norm bounds, a key component for verifying the sparsity of CKKS secret keys.

4.1 Basic Settings

In this subsection, we summarize the basic settings used in the prior work [HLSS25]. In designing a ZKAoK for CKKS, we also follow its approach. Suppose we want to prove the well-formedness of a public key pk and a ciphertext ct defined over the polynomial ring R_Q for some integer $Q > 0$. The prior work utilizes the following two observations in the design of a ZKAoK for BFV.

1. If $\left\lfloor \frac{p}{q}\mathsf{pk} \right\rceil$ and $\left\lfloor \frac{p}{q}\mathsf{ct} \right\rceil$ are well-formed in R_p, then pk and ct are also well-formed in R_Q, provided that $p \approx Q$.
2. If p is a prime such that $p = 1 \pmod{2N}$, then $R_p \cong \mathbb{Z}_p^N$.

Based on these observations, the prior work first translates the well-formedness conditions defined over R_Q into constraints defined over $\mathbb{Z}_p^N$, and then utilizes PIOPs designed for $\mathbb{Z}_p^N$ to prove the translated constraints. To accomplish this, elements in R_p must be represented in $\mathbb{Z}_p^N$. For this purpose, the following two vector representations are used.

- $\underline{\texttt{Coeff}}(\boldsymbol{a}) \to \vec{a}$: Given a polynomial $\boldsymbol{a} = \sum_{i=0}^{N-1} a_i X^i$, outputs a vector $\vec{a} = (a_0, \ldots, a_{N-1})$.
- $\underline{\texttt{NTT}}(\boldsymbol{a}) \to \vec{\underline{a}}$: Given a polynomial $\boldsymbol{a} = \sum_{i=0}^{N-1} a_i X^i$, outputs a vector $\vec{\underline{a}} = \big(\boldsymbol{a}(\xi), \boldsymbol{a}(\xi^3), \ldots, \boldsymbol{a}(\xi^{2N-1})\big)$, where ξ is a $2N$-th root of unity in $\mathbb{Z}_p$.

The first representation is simply a translation of coefficients; thus, if we prove any norm-related constraints on this representation of a witness polynomial, it corresponds to proving norm-related constraints on the witness polynomial itself. The second representation is an evaluation of the isomorphism $R_p \cong \mathbb{Z}_p^N$, known as the number-theoretic transform (NTT). As this is an isomorphism, proving arithmetic constraints on this representation of witness polynomials results in proving arithmetic constraints on the witness polynomials themselves. Additionally, there exist matrices $T, P_\varphi, P_\psi \in \mathbb{Z}_p^{N \times N}$ such that

$$\texttt{NTT}(\boldsymbol{a}) = T \cdot \texttt{Coeff}(\boldsymbol{a}), \quad \texttt{NTT}(\varphi(\boldsymbol{a})) = P_\varphi \cdot \texttt{NTT}(\boldsymbol{a}), \quad \texttt{NTT}(\psi(\boldsymbol{a})) = P_\psi \cdot \texttt{NTT}(\boldsymbol{a})$$

for $\boldsymbol{a} \in R_p$. Based on these properties, the prior work constructs a ZKAoK for BFV using PIOPs that prove constraints in $\mathbb{Z}_p^N$.

4.2 Encoding Method

In this subsection, we present an encoding method used in [HLSS25]. Once witnesses in R_p are represented as vectors in $\mathbb{Z}_p^N$, these vectors must be encoded into polynomials in $\mathbb{Z}_p[X]$, since the definition of a PIOP requires that all messages from the prover are in the form of polynomial oracles. Thus, to prove constraints over $\mathbb{Z}_p^N$ in PIOPs, the following encoding methods are used, which transform a vector into a polynomial via interpolation. For the interpolation, a multiplicative subgroup $\mathbb{H} = \{h_0, \ldots, h_{N-1}\} \subset \mathbb{Z}_p^\times$ of order N is typically used.

- $\underline{\texttt{Ecd}}(\vec{w}) \to \boldsymbol{w}$: Given a vector $\vec{w} \in \mathbb{Z}_p^N$, output the polynomial $\boldsymbol{w} \in \mathbb{Z}_p^{<N}[X]$ such that $\boldsymbol{w}(h_i) = w_i$ for $0 \leq i < N$.
- $\underline{\texttt{REcd}}(\vec{w}) \to \hat{\boldsymbol{w}}$: Given a vector $\vec{w} \in \mathbb{Z}_p^N$, uniformly sample a polynomial $\hat{\boldsymbol{w}} \in \mathbb{Z}_p^{<2N}[X]$ such that $\hat{\boldsymbol{w}}(h_i) = w_i$ for $0 \leq i < N$.
- $\underline{\texttt{Dcd}}(\boldsymbol{w}) \to \vec{w}$: Given a polynomial $\boldsymbol{w} \in \mathbb{Z}_p[X]$, output a vector $\vec{w} = (\boldsymbol{w}(h_0), \ldots, \boldsymbol{w}(h_{N-1}))$.

For $\vec{w} \in \mathbb{Z}_p^N$, there exists a unique polynomial $\boldsymbol{w} \in \mathbb{Z}_p^{<N}[X]$ corresponding to $\texttt{Ecd}(\vec{w})$. However, evaluating $\boldsymbol{w}$ at any point reveals information about $\vec{w}$, potentially compromising the zero-knowledge property in PIOPs. In contrast, for $\hat{\boldsymbol{w}}$, multiple valid candidates exist, allowing us to choose one at random. As a result, for $\alpha \notin \mathbb{H}$, the evaluation $\hat{\boldsymbol{w}}(\alpha)$ remains independent of $\vec{w}$ for up to $N-1$ evaluations, due to the bounded independence property. For further details, see [BSCR+19]. Once we encode vectors into polynomials using the above encoding methods, we can leverage the following properties to prove constraints in the vector space $\mathbb{Z}_p^N$.

Lemma 1 (Univariate Sum Check [BSCR+19]). *Let $\boldsymbol{f} \in \mathbb{Z}_p[X]$, $\mathbb{H} \subseteq \mathbb{Z}_p^\times$ be a multiplicative subgroup of order N, and $\boldsymbol{z}_\mathbb{H} = \prod_{h \in \mathbb{H}}(X - h) = (X^N - 1)$. Then, $\sum_{h \in \mathbb{H}} \boldsymbol{f}(h) = \mu$ holds if and only if there exist polynomials $\boldsymbol{q} \in \mathbb{Z}_p[X]$ and $\boldsymbol{r} \in \mathbb{Z}_p^{<N-1}[X]$ such that $\boldsymbol{f} = \boldsymbol{q} \cdot \boldsymbol{z}_\mathbb{H} + \boldsymbol{r} \cdot X + N^{-1} \cdot \mu \pmod p$ holds.*

Lemma 2 (Univariate Zero Test [BSCR+19]). *Let $\boldsymbol{f} \in \mathbb{Z}_p[X]$, $\mathbb{H} \subseteq \mathbb{Z}_p^\times$ be a multiplicative subgroup of order N, and $\boldsymbol{z}_\mathbb{H} = \prod_{h \in \mathbb{H}}(X - h) = (X^N - 1)$. Then, $\boldsymbol{f}(h) = 0$ for all $h \in \mathbb{H}$ if and only if there exists a polynomial $\boldsymbol{q} \in \mathbb{Z}_p[X]$ such that $\boldsymbol{f} = \boldsymbol{q} \cdot \boldsymbol{z}_\mathbb{H} \pmod p$ holds.*

4.3 PIOPs from the Prior Work

In this subsection, we review the PIOPs from the prior work [HLSS25,BSCR+19], which provide basic building blocks for proving constraints in $\mathbb{Z}_p^N$. For simplicity, we describe only their core functionality here. Complete protocol descriptions are provided in Appendix C.

PIOP for Linear Relations. For witness vectors $\vec{a}_i, \vec{b}_i \in \mathbb{Z}_p^N$ satisfying the linear relations $\vec{b}_i = M_i \vec{a}_i$ for public matrices $M_i \in \mathbb{Z}_p^{N \times N}$, where $0 \leq i < k$, their validity can be proven using the following PIOP.

$$\Pi_{\mathtt{Lin}}(M_0, \ldots, M_{k-1}; [\![\hat{\boldsymbol{a}}_0]\!], \ldots, [\![\hat{\boldsymbol{a}}_{k-1}]\!]; [\![\hat{\boldsymbol{b}}_0]\!], \ldots, [\![\hat{\boldsymbol{b}}_{k-1}]\!])$$

Instance: matrices $M_i \in \mathbb{Z}_p^{N \times N}$ for $0 \leq i < k$.
Witness: vectors $\vec{a}_i = \mathtt{Dcd}(\hat{\boldsymbol{a}}_i)$ and $\vec{b}_i = \mathtt{Dcd}(\hat{\boldsymbol{b}}_i)$ for $0 \leq i < k$, where $\hat{\boldsymbol{a}}_i, \hat{\boldsymbol{b}}_i \in \mathbb{Z}_p^{\leq 2N}[X]$.
Statement: $\vec{b}_i = M_i \vec{a}_i$ for $0 \leq i < k$.

Fig. 1: Functionality of the PIOP for batched linear relations

PIOP for Arithmetic Constraints. An arithmetic constraint over $\mathbb{Z}_p^N$, where addition and multiplication are component-wise, can be represented as a multivariate polynomial $\vec{\mathtt{C}}$ over the product ring $\mathbb{Z}_p^N$. The satisfiability of $\vec{\mathtt{C}}$ for witness vectors in $\mathbb{Z}_p^N$ can be verified using the following PIOP.

$$\Pi_{\mathtt{AC}}(\vec{\mathtt{C}}; [\![\hat{\boldsymbol{a}}_0]\!], \ldots, [\![\hat{\boldsymbol{a}}_{k-1}]\!])$$

Instance: k-ary circuit $\vec{\mathtt{C}} \in (\mathbb{Z}_p^N)[\vec{X}_0, \ldots, \vec{X}_{k-1}]$ of degree d.
Witness: vectors $\vec{a}_i = \mathtt{Dcd}(\boldsymbol{a}_i)$ for $0 \leq i < k$, where $\hat{\boldsymbol{a}}_i \in \mathbb{Z}_p^{\leq 2N}[X]$.
Statement: $\vec{\mathtt{C}}(\vec{a}_0, \ldots, \vec{a}_{k-1}) = 0$.

Fig. 2: Functionality of the PIOP for arithmetic constraints

PIOP for L^∞-Norm Constraints. For witness vectors $\vec{a}_0, \ldots, \vec{a}_{k-1} \in \mathbb{Z}_p^N$, one can prove their L^∞-norm upper bound in a batched manner using the following PIOP.

$$\Pi_{L^\infty}(B; [\![\hat{\boldsymbol{a}}_0]\!], \ldots, [\![\hat{\boldsymbol{a}}_{k-1}]\!])$$

Instance: norm bound B.
Witness: vectors $\vec{a}_i = \mathtt{Dcd}(\hat{\boldsymbol{a}}_i)$, where $\hat{\boldsymbol{a}}_i \in \mathbb{Z}_p^{\leq 2N}[X]$ for $0 \leq i < k$.
Statement: $\|\vec{a}_i\|_\infty \leq B$ for $0 \leq i < k$.

Fig. 3: Functionality of the PIOP for L^∞-norm constraints

4.4 Our PIOP for L^2-Norm Constraints

We now present our new PIOP for proving L^2-norm constraints. We begin by generalizing the univariate sumcheck protocol from [BSCR+19], which verifies the relation $\sum_{h \in \mathbb{H}} \boldsymbol{a}(h) = 0$ for a committed polynomial $\boldsymbol{a}$. In the context of polynomial encoding, this condition is equivalent to the inner product relation $\left\langle \vec{a}, \vec{1} \right\rangle = 0$, where $\vec{a} = \mathtt{Dcd}(\boldsymbol{a})$. However, to prove upper bounds on the L^2-norm, we require a more general form of this inner product relation. Specifically, instead

of $\vec{a}$, we consider $\vec{\mathtt{C}}(\vec{a}_0, \ldots, \vec{a}_{k-1})$, where $\vec{\mathtt{C}}$ is a multivariate polynomial over the product ring $\mathbb{Z}_p^N$, and $\vec{a}_0, \ldots, \vec{a}_{k-1} \in \mathbb{Z}_p^N$. Thanks to polynomial encoding, this generalized relation can still be verified using Lemma 1. Below, we extend the univariate sumcheck protocol from [BSCR+19] to prove the inner product relation $\left\langle \vec{\mathtt{C}}(\vec{a}_0, \ldots, \vec{a}_{k-1}), \vec{1} \right\rangle = 0$ for witness vectors $\vec{a}_0, \ldots, \vec{a}_{k-1}$.

Theorem 3. *Let* $\hat{\boldsymbol{a}}_i \leftarrow \mathtt{REcd}(\vec{a}_i)$ *for* $0 \le i < k$*, where* $\left\langle \vec{\mathtt{C}}(\vec{a}_0, \ldots, \vec{a}_{k-1}), \vec{1} \right\rangle = 0$*. Then, an interactive protocol* $\Pi_{\mathtt{IP}}$ *described in Fig. 4 is an HVZK PIOP with a soundness error of* $\frac{O(dN)}{p-N}$*.*

Proof. See Appendix D.1. □

$$\Pi_{\mathtt{IP}}(\vec{\mathtt{C}}; [\![\hat{\boldsymbol{a}}_0]\!], \ldots, [\![\hat{\boldsymbol{a}}_{k-1}]\!])$$

Instance: k-ary circuit $\vec{\mathtt{C}} \in (\mathbb{Z}_p^N)[\vec{X}_0, \ldots, \vec{X}_{k-1}]$ of degree d.
Witness: vectors $\vec{a}_i = \mathtt{Dcd}(\hat{\boldsymbol{a}}_i)$ for $0 \le i < k$, where $\hat{\boldsymbol{a}}_i \in \mathbb{Z}_p^{\le 2N}[X]$.
Statement: $\left\langle \vec{\mathtt{C}}(\vec{a}_0, \ldots, \vec{a}_{k-1}), \vec{1} \right\rangle = 0$.

1. Both $\mathtt{P}$ and $\mathtt{V}$ compute $\mathbf{C} \in (\mathbb{Z}_p[X])[\boldsymbol{X}_0, \ldots, \boldsymbol{X}_{k-1}]$ of degree d, which is obtained by applying $\mathtt{Ecd}$ to each coefficient of $\vec{\mathtt{C}}$.
2. The prover $\mathtt{P}$ samples a random polynomial $\boldsymbol{g}$ of degree $\le d(2N-1)+N-1$, computes the summation $\mu = \sum_{h \in \mathbb{H}} \boldsymbol{g}(h)$, and sends polynomial oracles $([\![\boldsymbol{g}]\!], d(2N-1)+N-1)$ and μ to the verifier $\mathtt{V}$.
3. The verifier $\mathtt{V}$ sends a random point $\beta \leftarrow \mathcal{U}(\mathbb{Z}_p \setminus \mathbb{H})$.
4. The prover $\mathtt{P}$ computes $\boldsymbol{q}$ and $\boldsymbol{r}$, which satisfy the following.

$$\boldsymbol{q} \cdot \boldsymbol{z}_{\mathbb{H}} + \boldsymbol{r} \cdot X + N^{-1} \cdot \mu = \boldsymbol{g} + \beta \cdot \mathbf{C}(\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_{k-1})$$

   Then, $\mathtt{P}$ sends polynomial oracles $([\![\boldsymbol{q}]\!], d(2N-1)-1)$ and $([\![\boldsymbol{r}]\!], N-2)$ to the verifier $\mathtt{V}$.
5. The verifier $\mathtt{V}$ gets evaluations $\hat{\boldsymbol{a}}_i(\alpha)$, $\boldsymbol{g}(\alpha)$, $\boldsymbol{q}(\alpha)$, and $\boldsymbol{r}(\alpha)$ by accessing polynomial oracles $[\![\hat{\boldsymbol{a}}_i]\!]$, $[\![\boldsymbol{g}]\!]$, $[\![\boldsymbol{q}]\!]$, and $[\![\boldsymbol{r}]\!]$ at a random point $\alpha \leftarrow \mathcal{U}(\mathbb{Z}_p \setminus \mathbb{H})$. Then, $\mathtt{V}$ checks whether the following holds.

$$\boldsymbol{q}(\alpha) \cdot \boldsymbol{z}_{\mathbb{H}}(\alpha) + \boldsymbol{r}(\alpha) \cdot \alpha + N^{-1} \cdot \mu = \boldsymbol{g}(\alpha) + \beta \cdot \big(\mathbf{C}(\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_{k-1})\big)(\alpha)$$

Fig. 4: PIOP for an inner product relation

Based on the protocol $\Pi_{\mathtt{IP}}$ for inner product relations, we now present a new PIOP for proving upper bounds on the L^2-norm of witness vectors in $\mathbb{Z}_p^N$. We start with the observation that $\|\vec{a}\|_2^2 = \langle \vec{a}, \vec{a} \rangle$, so it suffices to prove the range of $\langle \vec{a}, \vec{a} \rangle$. However, since all arithmetic is performed in $\mathbb{Z}_p$, what we can actually prove is the range of $[\langle \vec{a}, \vec{a} \rangle]_p$. To address this issue, we use Π_{L^∞}, which proves an upper bound on $\|\vec{a}\|_\infty$. Specifically, the condition $\|\vec{a}\|_\infty \le B_\infty$ for some $B_\infty > 0$ implies $\|\vec{a}\|_2 \le B_\infty^2 N$. Therefore, unless $B_\infty^2 N > p$, we can ensure that $[\langle \vec{a}, \vec{a} \rangle]_p = \langle \vec{a}, \vec{a} \rangle$. Below, we describe our PIOP that proves the range of

the L^2-norm modulo p, which can handle multiple witness vectors in a batched manner.

Theorem 4. *Let* $\hat{\boldsymbol{a}}_i \leftarrow \mathtt{REcd}(\vec{a}_i)$ *, where* $0 \le [\|\vec{a}_i\|_2^2]_p \le B$ *for* $0 \le i < k$*. Then, an interactive protocol* Π_{L^2} *described in Fig. 5 is an HVZK PIOP with a soundness error of* $\frac{O(k+N)}{p-N}$*.*

Proof. See Appendix D.2. □

$$\Pi_{L^2}(B; [\![\hat{\boldsymbol{a}}_0]\!], \cdots, [\![\hat{\boldsymbol{a}}_{k-1}]\!])$$

Instance: norm bound B
Witness: vectors $\vec{a}_i = \mathtt{Dcd}(\boldsymbol{a}_i)$, where $\boldsymbol{a}_i \in \mathbb{Z}_p^{<2N}[X]$ for $0 \le i < k$
Statement: $0 \le \left[\|\vec{a}_i\|_2^2\right]_p \le B$ for $0 \le i < k$.

1. The prover $\mathtt{P}$ and the verifier $\mathtt{V}$ decompose B into $B_0 = \left\lfloor \frac{B}{2} \right\rceil$, $B_1 = \left\lfloor \frac{B-B_0}{2} \right\rceil$, $B_2 = \left\lfloor \frac{B-B_0-B_1}{2} \right\rceil, \ldots, B_{\ell-1} = 1$, where $\ell = \lfloor \log B \rfloor + 1$, and set $\vec{b} = (B_0, \ldots, B_{\ell-1}, 0, \ldots, 0)$ and $\vec{c} = (1, \ldots, 1, 0, \ldots, 0)$, where the first ℓ elements are nonzero.
2. The prover $\mathtt{P}$ computes $u_{i,0}, \ldots, u_{i,\ell-1} \in \{0,1\}$ such that $\langle \vec{a}_i, \vec{a}_i \rangle = \sum_{j=0}^{\ell-1} u_{i,j} \cdot B_i \pmod p$ for $0 \le i < k$. Then, it samples $\hat{\boldsymbol{u}}_i \leftarrow \mathtt{REcd}(u_{i,0}, \ldots, u_{i,\ell-1}, 0, \ldots, 0)$, and sends the polynomial oracle $([\![\hat{\boldsymbol{u}}_i]\!], 2N-1)$ to the verifier $\mathtt{V}$ for $0 \le i < k$.
3. The verifier sends a random point $\gamma \leftarrow \mathcal{U}(\mathbb{Z}_p \setminus \mathbb{H})$
4. The prover $\mathtt{P}$ and the verifier $\mathtt{V}$ invoke the following PIOP.

$$\Pi_{\mathtt{AC}}\left(\sum_{i=0}^{k-1} \gamma^i \cdot \vec{X}_i \odot (\vec{X}_i - \vec{c}); \{[\![\hat{\boldsymbol{u}}_i]\!]\}_{0 \le i < k}\right)$$

$$\Pi_{\mathtt{IP}}\left(\sum_{i=0}^{k-1} \gamma^i \cdot (\vec{X}_i^2 - \vec{b} \odot \vec{X}_{i+k}); \{[\![\hat{\boldsymbol{a}}_i]\!]\}_{0 \le i < k}, \{[\![\hat{\boldsymbol{u}}]\!]\}_{0 \le i < k}\right)$$

Fig. 5: PIOP for L^2-norm constraints

5 Secure Function Evaluation via CKKS

In this section, we discuss the security and privacy of secure function evaluation protocols based on the CKKS scheme. We focus on the setting where the sender is semi-honest and the receiver may be malicious, because in a HE-based protocol, the sender receives the receiver's ciphertext and public key only in the first round. Thus, for malicious senders, there is no opportunity to compromise the receiver's privacy using only these materials.

We begin by introducing our new security definition, which we refer to as differentially private homomorphic evaluation protocols. Then, we present a general compilation method based on the Laplace mechanism that achieves sender privacy for any plain CKKS-based protocol, assuming the existence of an ideal

functionality for ZKAoK of CKKS ciphertexts and public keys. Finally, we describe how to instantiate a ZKAoK for CKKS by devising a new proof technique that combines the PIOPs introduced in Section 4 with the coeff-to-slot homomorphic operation.

5.1 Differentially Private Homomorphic Evaluation Protocol

We begin by presenting the basic structure of CKKS-based secure function evaluation protocols, which we refer to as a *homomorphic evaluation* protocol. The formal definition is as follows.

Definition 9 (Homomorphic Evaluation). *Let* $(\mathtt{P}_1, \mathtt{P}_2)$ *be a sender-receiver protocol, and let* $\mathtt{f} : \Theta \times \mathcal{X}^k \to \mathcal{Y}$ *be a deterministic algorithm, where* $\mathcal{X}, \mathcal{Y} \subseteq R$*. We say that* $(\mathtt{P}_1, \mathtt{P}_2)$ *is a homomorphic evaluation protocol for* $\mathtt{f}$ *if it has the following structure.*

Inputs. *The receiver* $\mathtt{P}_1$ *has a message* $\vec{\boldsymbol{x}} \in \mathcal{X}^k$ *as a private input. The sender* $\mathtt{P}_2$ *has a parameter* $\theta \in \Theta$ *as a private input. A public parameter* $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$ *is public to both parties.*

The protocol.

1. *The receiver* $\mathtt{P}_1$ *sends* $(\mathsf{pk}, \mathsf{ct}_0, \ldots, \mathsf{ct}_{k-1})$ *to the sender* $\mathtt{P}_2$*, where* pk *is a public key, and* $\{\mathsf{ct}_i\}_{0 \le i < k}$ *are ciphertext.*
2. *The receiver* $\mathtt{P}_1$ *and the sender* $\mathtt{P}_2$ *invoke auxiliary subprotocols that are independent of* θ*.*
3. *The sender* $\mathtt{P}_2$ *computes a ciphertext* $\mathsf{ct}_{\mathsf{out}}$*, and sends it to the receiver* $\mathtt{P}_1$*.*
4. *The receiver* $\mathtt{P}_1$ *outputs* $\boldsymbol{y} = \mathtt{Dec}_{\mathsf{sk}}(\mathsf{ct}_{\mathsf{out}}) \in \mathcal{Y}$*.*

Next, we provide the definition of *differentially private prediction* [DF18] with a slight relaxation for computationally bounded adversaries. This serves as a baseline for defining sender privacy in homomorphic evaluation protocols.

Definition 10 ([DF18, DP Prediction]). *Let* $\mathtt{M} : \Theta \times \mathcal{X}^k \to \mathcal{Y}$ *be a randomized algorithm. We say that* $\mathtt{M}$ *is an* ε*-DP prediction algorithm if, for every* $\vec{\boldsymbol{x}} \in \mathcal{X}^k$*, the output* $\mathtt{M}(\theta, \vec{\boldsymbol{x}})$ *is* ε*-CDP with respect to* θ*.*

Based on the definition of DP prediction, we define our security notion for homomorphic evaluation protocols, where we model the ideal functionality of the protocol as a DP prediction.

Definition 11 (Differentially Private Homomorphic Evaluation). *Let* $\mathtt{f} : \Theta \times \mathcal{X}^k \to \mathcal{Y}$ *be a deterministic algorithm, where* $\mathcal{X}, \mathcal{Y} \subseteq R$*, let* $\mathtt{M} : \Theta \times \mathcal{X}^k \to \mathcal{Y}$ *be a randomized algorithm, and let* $\Pi = (\mathtt{P}_1, \mathtt{P}_2)$ *be a homomorphic evaluation protocol. We say that* Π *is an* ε*-differentially private homomorphic evaluation (DPHE) protocol for* $\mathtt{f}$ *with* (η, δ)*-usefulness if the following holds.*

Usefulness. *For all* $\vec{\boldsymbol{x}} \in \mathcal{X}^k$ *and* $\theta \in \Theta$*, the receiver's output* $\boldsymbol{y}$ *from the correct execution of* Π *with inputs* $\vec{\boldsymbol{x}}$ *and* θ *satisfies the following condition.*

$$\Pr\left[\|\boldsymbol{y} - \mathtt{f}(\theta, \vec{\boldsymbol{x}})\|_\infty > \eta\right] \le \delta$$

Receiver's privacy. *For all sender's input $\theta \in \Theta$, the views of the honest sender $\mathtt{P}_2$ for all receiver's inputs $\vec{\boldsymbol{x}}, \vec{\boldsymbol{x}}' \in \mathcal{X}^k$ are computationally indistinguishable. i.e.,* $\mathtt{View}^{\Pi}_{\mathtt{P}_2}(\vec{\boldsymbol{x}}, \theta) \approx_c \mathtt{View}^{\Pi}_{\mathtt{P}_2}(\vec{\boldsymbol{x}}', \theta)$.

Sender's privacy. *For all PPT adversaries* $\mathtt{A}$ *that only manipulate the receiver, there exists a PPT simulator* $\mathtt{S}$ *such that for all PPT environments* $\mathtt{Z}$ *and PPT algorithms* $\mathtt{D}$*, the following holds for all $\theta \in \Theta$, where Π_θ is a restriction of Π with the sender's input θ, and $\mathtt{F}_\theta$ is the ideal functionality defined in Fig. 6.*

$$\Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_\theta, \mathtt{A}, \mathtt{Z}]) = 1\right] \leq e^{\varepsilon} \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}, \mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda)$$

$$\Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}, \mathtt{Z}]) = 1\right] \leq e^{\varepsilon} \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_\theta, \mathtt{A}, \mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda)$$

$\mathtt{F}_\theta$

Inputs: The receiver $\mathtt{P}_1$ sends $\vec{\boldsymbol{x}} \in \mathcal{X}^k$.
Outputs: The receiver $\mathtt{P}_1$ obtains $\boldsymbol{y} \leftarrow \mathtt{M}(\theta, \vec{\boldsymbol{x}})$.

Fig. 6: Ideal functionality for differentially private homomorphic evaluation

Since the ideal functionality in our definition is a randomized algorithm, we relax the correctness condition to a notion of usefulness, which describes how far the output may deviate from the target. For the receiver's privacy, we consider only a semi-honest sender and adopt the standard MPC definition based on computational indistinguishability. However, for the sender's privacy, we consider the possibility of a malicious receiver. Thus, we incorporate the UC framework [Can01] alongside the notion of computational differential privacy.

Next, we present the main implication of our DPHE protocol, which states that if the ideal functionality provides a DP prediction and the protocol satisfies the sender's privacy condition, then the output of the real protocol execution also satisfies the DP prediction property. We describe it formally as follows.

Theorem 5 (Composition of CDP and DPHE). *Let* $\mathtt{f} : \Theta \times \mathcal{X}^k \to \mathcal{Y}$ *be a deterministic algorithm, and let* Π *be a ε_1-DPHE protocol for* $\mathtt{f}$*. Suppose* $\mathtt{M}$ *is a ε_2-DP prediction with respect to θ. Then,* $\mathtt{EXEC}[\Pi_\theta, \mathtt{A}, \mathtt{Z}]$ *is a $(2\varepsilon_1 + \varepsilon_2)$-DP prediction with respect to θ for all PPT environments* $\mathtt{Z}$ *and all PPT adversaries* $\mathtt{A}$ *that manipulate the receiver.*

Proof. See Appendix D.3. □

Based on the above theorem, existing DP prediction protocols can be naturally extended to DPHE protocols, provided that the underlying homomorphic evaluation protocol satisfies the DPHE property.

Now, we examine whether the commonly used proof techniques in the UC framework based on computational indistinguishability are well suited to our differential privacy setting. First, we show that the completeness result for the dummy adversary is indeed well suited to our setting, which allows us to consider only the dummy adversary when proving the protocol's security in the UC

framework. We then prove that our definition also supports subprotocol composition, enabling the establishment of protocol security in the hybrid execution model. Below, we present the adaptations of these proof techniques within our security definition.

Theorem 6 (Completeness of the dummy adversary). *Let Π be a homomorphic evaluation protocol such that the sender's privacy holds for the dummy adversary $\mathtt{A}^*$, whose only job is to forward messages between an environment and the other protocol parties, with a parameter ε for all PPT environments. Then, Π satisfies the sender's privacy with the parameter ε.*

Proof. See Appendix D.4. □

Theorem 7 (Subprotocol composition). *Let Π be a homomorphic evaluation protocol that uses Π' as a subprotocol, which securely implements an ideal functionality $\mathtt{F}'$, and let $\Pi_{\mathtt{F}'}$ be the hybrid protocol, which uses the ideal functionality $\mathtt{F}'$ in place of Π'. If $\Pi_{\mathtt{F}'}$ satisfies the sender's privacy with a parameter ε, then Π also satisfies the sender's privacy with the parameter ε.*

Proof. See Appendix D.5. □

5.2 DPHE Compiler from ZKAoK for CKKS

In this subsection, we present a general compilation method to transform a plain homomorphic evaluation protocol into a DPHE protocol. Our approach is inspired by [AJLA+12], which provides a general compilation method based on noise flooding and ZKAoK for BGV. First, we replace the noise flooding method, which introduces exponentially large noise, with the Laplace mechanism, which adds moderate-sized noise to conceal sender information. We then define the ideal functionality of a ZKAoK for CKKS that ensures the correctness of the Laplace mechanism when applied to CKKS ciphertexts. As an additional optimization, we achieve ciphertext rerandomization based on the Hint-MLWE problem, which removes sender-specific information embedded in the ciphertext by adding an encryption of zero. In prior work, this rerandomization is also achieved via noise flooding, which incurs significant parameter overhead. However, our method avoids this overhead by employing a Hint-MLWE-based analysis. Below, we describe our compilation method in detail.

Theorem 8. *Let $\mathtt{f} : \Theta \times \mathcal{X}^k \to \mathcal{Y}^k$ be a deterministic algorithm, where $\mathtt{f}(\theta, \cdot)$ is admissible in CKKS for all $\theta \in \Theta$. Let $\Pi_{\mathtt{CKKS}}$ be the homomorphic evaluation protocol described in Fig. 7, and let $\mathtt{M}(\theta, \vec{\boldsymbol{x}}) = \mathtt{f}(\theta, \vec{\boldsymbol{x}}) + \lfloor \boldsymbol{t} \rceil$ for $\boldsymbol{t} \leftarrow \mathtt{Lap}(N\tau/\varepsilon)^N$. Then $\Pi_{\mathtt{CKKS}}$ is an ε-DPHE protocol for $\mathtt{f}$ in the $\mathtt{F}_{\mathtt{ZKAoK}}$-hybrid model, which is described in Fig. 8, if we assume $\mathsf{RLWE}_{Q_1,\sigma/2,\sigma/2}$ is computationally hard, and $\sigma \geq 2\sqrt{2} \cdot \eta_{(1/2^\lambda)}(\mathbb{Z}^N)$.*

Proof. See Appendix D.6. □

$\Pi_{\mathtt{CKKS}}$

Inputs: The receiver $\mathtt{P}_1$ has a message $\vec{\boldsymbol{x}} \in \mathcal{X}^k$ as a private input. The sender $\mathtt{P}_2$ has a parameter $\theta \in \Theta$ as a private input. A public parameter $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$ is public to both parties.

The protocol:

1. The receiver $\mathtt{P}_1$ generates the followings, and sends $(\mathsf{pk}, \mathsf{ct}_0, \ldots, \mathsf{ct}_{k-1})$ to the sender $\mathtt{P}_2$.
$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{KeyGen}(\mathsf{pp}),\ \mathsf{ct}_i \leftarrow \mathtt{Enc}_{\mathsf{sk}}(\boldsymbol{x}_i)$$
2. The parties invoke $\mathtt{F}_{\mathtt{ZKAoK}}$, and the sender $\mathtt{P}_2$ aborts if its output is 0.
3. The sender $\mathtt{P}_2$ computes the following.
$$(\boldsymbol{c}_0, \boldsymbol{c}_1) \leftarrow \left[\mathtt{Eval}_{\mathsf{pk}}(\mathtt{f}(\theta;\cdot); \mathsf{pk}, \mathsf{ct}_0, \ldots, \mathsf{ct}_{k-1})\right]_{Q_0}$$
$$\tau = \max_{\theta \in \Theta} \left(\mathtt{Estim}\big(\mathtt{f}(\theta,\cdot); h, B_{\mathsf{rlk}}, B_{\mathsf{rtk}}, B_{\mathsf{cjk}}, B_e\big) \right)$$
The sender $\mathtt{P}_2$ sends an output ciphertext $\mathsf{ct}_{\mathsf{out}}$ as follows, and send $\mathsf{ct}_{\mathsf{out}}$ to the receiver $\mathtt{P}_2$.
$$\boldsymbol{r}_0 \leftarrow \mathtt{DG}(\sigma)^N, \boldsymbol{r}_1 \leftarrow \mathtt{DG}(B_{\mathsf{ek}} N \cdot \sigma)^N, \boldsymbol{r}_2 \leftarrow \mathtt{DG}(h\sigma)^N, \boldsymbol{r}_3 \leftarrow \mathtt{DG}(\sigma)^N, \boldsymbol{t} \leftarrow \mathtt{Lap}(N\tau/\varepsilon)^N$$
$$\mathsf{ct}_{\mathsf{out}} = \boldsymbol{r}_0 \cdot \mathsf{ek} + (\boldsymbol{r}_1 + \boldsymbol{r}_2, \boldsymbol{r}_3) + q_1 \cdot (\boldsymbol{c}_0 + \lfloor \boldsymbol{t} \rceil, \boldsymbol{c}_1) \pmod{Q_1}$$
4. The receiver obtains an output $\boldsymbol{y} = \mathtt{Dec}_{\mathsf{sk}}(\mathsf{ct}_{\mathsf{out}})$.

Fig. 7: Protocol for differentially private homomorphic evaluation using CKKS

$\mathtt{F}_{\mathtt{ZKAoK}}$

Inputs: The receiver $\mathtt{P}_1$ sends $(\mathsf{x}_{\mathtt{CKKS}}, \mathsf{w}_{\mathtt{CKKS}})$, where $\mathsf{x}_{\mathtt{CKKS}} = (\mathsf{pk}, \mathsf{ct}_0, \ldots, \mathsf{ct}_{k-1})$, $\mathsf{w}_{\mathtt{CKKS}} = (\boldsymbol{s}, \boldsymbol{e}_{\mathsf{ek}}, \vec{\boldsymbol{e}}_{\mathsf{rlk}}, \vec{\boldsymbol{e}}_{\mathsf{rtk}}, \vec{\boldsymbol{e}}_{\mathsf{cjk}}, \vec{\boldsymbol{e}}, \vec{\boldsymbol{x}})$, $\mathsf{ek} = (\mathsf{ek}_0, \mathsf{ek}_1)$, $\mathsf{rlk} = (\mathsf{rlk}_0, \mathsf{rlk}_1)$, $\mathsf{rtk} = (\mathsf{rtk}_0, \mathsf{rtk}_1)$, $\mathsf{cjk} = (\mathsf{cjk}_0, \mathsf{cjk}_1)$, $\vec{\boldsymbol{e}} = (\boldsymbol{e}_0, \ldots, \boldsymbol{e}_{k-1})$, $\vec{\boldsymbol{x}} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{k-1})$, and $\mathsf{ct}_i = (\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1})$ for $0 \le i < k$, and the sender $\mathtt{P}_2$ sends $\mathsf{x}_{\mathtt{CKKS}}$.

Outputs: The sender $\mathtt{P}_2$ obtains 1 if $(\mathsf{x}_{\mathsf{CKKS}}, \mathsf{w}_{\mathsf{CKKS}}) \in \mathsf{R}_{\mathsf{CKKS}}$, where

$$\mathsf{R}_{\mathsf{CKKS}} = \left\{ (\mathsf{x}_{\mathsf{CKKS}}, \mathsf{w}_{\mathsf{CKKS}}) \;\middle|\; \begin{array}{r} \|\boldsymbol{s}\|_\infty \le 1 \wedge \|\boldsymbol{s}\|_1 \le h \wedge \\ \|\boldsymbol{e}_{\mathsf{ek}}\|_\infty \le B_{\mathsf{ek}} \wedge \|\vec{\boldsymbol{e}}_{\mathsf{rlk}}\|_\infty \le B_{\mathsf{rlk}} \wedge \|\vec{\boldsymbol{e}}_{\mathsf{rtk}}\|_\infty \le B_{\mathsf{rtk}} \wedge \\ \|\vec{\boldsymbol{e}}_{\mathsf{cjk}}\|_\infty \le B_{\mathsf{cjk}} \wedge \|\vec{\boldsymbol{e}}\|_\infty \le B_e \wedge \\ \boldsymbol{x}_0, \ldots, \boldsymbol{x}_{k-1} \in \mathcal{X} \wedge \\ \mathsf{ek}_0 + \boldsymbol{s} \cdot \mathsf{ek}_1 = \boldsymbol{e}_{\mathsf{ek}} \wedge \\ \mathsf{rlk}_0 + \boldsymbol{s} \cdot \mathsf{rlk}_1 = \boldsymbol{s}^2 \cdot \vec{g} + \vec{\boldsymbol{e}}_{\mathsf{rtk}} \wedge \\ \mathsf{rtk}_0 + \boldsymbol{s} \cdot \mathsf{rtk}_1 = \varphi(\boldsymbol{s}) \cdot \vec{g} + \vec{\boldsymbol{e}}_{\mathsf{rtk}} \wedge \\ \mathsf{cjk}_0 + \boldsymbol{s} \cdot \mathsf{cjk}_1 = \psi(\boldsymbol{s}) \cdot \vec{g} + \vec{\boldsymbol{e}}_{\mathsf{cjk}} \wedge \\ \mathsf{ct}_{i,0} + \boldsymbol{s} \cdot \mathsf{ct}_{i,1} = \boldsymbol{x}_i + \boldsymbol{e}_i \text{ for } 0 \le i < k \end{array} \right\}$$

Otherwise, it obatins 0.

Fig. 8: Ideal functionality for zero-knowledge argument of knowledge for CKKS public keys and ciphertexts

Next, we discuss which algorithm $\mathtt{f}$ is suitable for DPHE protocols. In the context of DP prediction [DF18], the following condition is typically considered, which can be seen as a generalization of the bounded sensitivity notion in differential privacy.

Definition 12 ([DF18, Uniformly RO-Stable Prediction]). *An algorithm $\mathtt{f} : \Theta \times \mathcal{X}^K \to \mathcal{Y}$ is called a uniformly stable replace-one prediction with rate ξ if, for all adjacent $\theta, \theta' \in \Theta$ and any $\vec{\boldsymbol{x}} \in \mathcal{X}^k$, $\|\mathtt{f}(\theta, \vec{\boldsymbol{x}}) - \mathtt{f}(\theta', \vec{\boldsymbol{x}})\|_\infty \le \xi$ holds.*

If an algorithm $\mathtt{f}$ satisfies the above definition, we can easily construct a DP prediction algorithm using the Laplace mechanism. Moreover, as shown in [DF18, SSSSS10], several classes of convex optimization-based training algorithms, such as binary classification, regression, and density estimation, yield inference algorithms $\mathtt{f}$ that are uniformly RO-stable with respect to the training data. Below, we describe the implications of a DPHE protocol when $\mathtt{f}$ is uniformly RO-stable.

Theorem 9. *Let $\mathtt{f} : \Theta \times \mathcal{X}^k \to \mathcal{Y}$ be a uniformly RO-stable prediction with rate ξ. Then, $\mathtt{EXEC}[\Pi_{\mathsf{CKKS},\theta}, \mathtt{A}, \mathtt{Z}]$ is a $(2 + \xi/\tau)\varepsilon$-DP prediction with respect to θ for all PPT environments $\mathtt{Z}$ and all PPT adversaries $\mathtt{A}$ that manipulate the receiver.*

Proof. If $\mathtt{f}$ is uniformly stable with rate ξ, then $\mathtt{M}$ is a $(\xi/\tau)\varepsilon$-DP prediction due to the Laplace mechanism. Therefore, by Theorem 5, $\mathtt{EXEC}[\Pi_{\mathsf{CKKS},\theta}, \mathtt{A}, \mathtt{Z}]$ is a $(2 + \xi/\tau)\varepsilon$-DP prediction with respect to θ. □

5.3 Construction of ZKAoK for CKKS via PIOP

In this subsection, we present the construction of a ZKAoK for the relation $\mathsf{R}_{\mathsf{CKKS}}$ using PIOPs. Our construction efficiently addresses CKKS-specific issues: proving the sparsity of the secret key and verifying the validity of the plaintext.

For the sparsity condition, we construct a proof by employing our newly designed PIOP Π_{L^2} for L^2-norm bounds. We note that, under the condition $\|\vec{s}\|_\infty \leq 1$, the inequality $[\|\vec{s}\|_2^2]_p \leq h$ implies $\|\vec{s}\|_1 = h$. Therefore, executing Π_{L^2} in conjunction with Π_{L^∞} allows us to prove a bound on the Hamming weight of the secret key.

To prove the validity of the plaintext, using PIOPs alone is insufficient, as we must prove that $\boldsymbol{x}_i \in \mathcal{X}$ for $0 \leq i < k$, which typically involves constraints defined over $\mathbb{C}^{N/2}$. To be precise, in most PPML applications of CKKS, the condition $\boldsymbol{x}_i \in \mathcal{X}$ is expressed as $\mathtt{Unpack}(\boldsymbol{x}_i) \in \prod_{j=0}^{N/2-1} \cup_{k=0}^{\ell-1} [a_{j,k}, b_{j,k}]$ for some $a_{j,k}, b_{j,k} \in \mathbb{R}$; that is, the input plaintext is unpacked to a vector whose components lie in a finite union of closed intervals in $\mathbb{R}$. This is because PPML applications require the evaluation of nonlinear functions such as the sine function [LLL+21], step function [CKK20], and softmax function [CHK+24], and these are usually performed via polynomial approximations of the respective functions, whose correctness is guaranteed only on certain finite unions of closed intervals.

Thus, we focus on the most basic and commonly used setting, where $\boldsymbol{x}_i \in \mathcal{X}$ stands for $\mathtt{Unpack}(\boldsymbol{x}_i) \in [-1, 1]^{N/2}$, i.e., $\|\mathtt{Unpack}(\boldsymbol{x}_i)\|_\infty \leq 1$ for $0 \leq i < k$. This is because, if we can prove this basic setting, it can be naturally extended to more general cases. In proving the condition $\|\mathtt{Unpack}(\boldsymbol{x}_i)\|_\infty \leq 1$, our key idea is to incorporate a homomorphic operation called the coeff-to-slot operation, which is a key building block for CKKS bootstrapping [CHK+18a]. Its functionality is described below.

- $\underline{\mathtt{CtoS}}(\mathsf{ct}) \to \mathsf{ct}', \mathsf{ct}''$: Given a CKKS ciphertext $\mathsf{ct} \in R^2_{Q_\ell}$, whose plaintext is $\mathtt{Dec}_{\mathsf{sk}}(\mathsf{ct}) = \boldsymbol{m} = \sum_{i=0}^{N-1} m_i X^i$, it outputs two ciphertexts $\mathsf{ct}', \mathsf{ct}'' \in R^2_{Q_{\ell-\ell_{\mathtt{CtoS}}}}$, whose plaintexts are $\boldsymbol{m}', \boldsymbol{m}'' \in R$, where $\iota(\boldsymbol{m}') = (m_0, \ldots, m_{N/2-1})$, and $\iota(\boldsymbol{m}'') = (m_{N/2}, \ldots, m_{N-1})$. [2]

Based on the coeff-to-slot operation, we modify the ciphertext generation process as follows. Given input messages $\vec{m}_0, \ldots, \vec{m}_{k-1} \in \mathbb{R}^{N/2}$, which are intended to be transformed into plaintexts via $\mathtt{Pack}$, we instead generate plaintexts $\boldsymbol{m}_i$, where $\mathtt{Coeff}(\boldsymbol{m}_i) = \lfloor \Delta \cdot (\vec{m}_{2i} \| \vec{m}_{2i+1}) \rceil$ for $0 \leq i < k/2$, and then generate input ciphertexts based on these plaintexts. During the ZKAoK, we prove that $\|\boldsymbol{m}_i\|_\infty \leq \Delta$ using Π_{L^∞}. Finally, applying $\mathtt{CtoS}$ to the verified ciphertexts results in CKKS ciphertexts whose plaintexts satisfy the previously described condition. Below, we formalize this approach.

Theorem 10. *Let* $(\mathsf{pk}', \mathsf{ct}'_0, \ldots, \mathsf{ct}'_{k/2-1}) \in \mathsf{L}(\mathsf{R}'_{\mathtt{CKKS}})$*, and let* $\mathsf{pk} = \left\lfloor \frac{Q_L}{p} \cdot \mathsf{pk}' \right\rceil$*, and* $(\mathsf{ct}_{2i}, \mathsf{ct}_{2i+1}) \leftarrow \mathtt{CtoS}_{\mathsf{pk}}\left(\left\lfloor \frac{Q_L}{p} \cdot \mathsf{ct}'_i \right\rceil \right)$ *for* $0 \leq i < k/2$ *and* $Q_L < p$*. Then, it holds that* $(\mathsf{pk}, \mathsf{ct}_0, \ldots, \mathsf{ct}_{k-1}) \in \mathsf{L}(\mathsf{R}_{\mathtt{CKKS}})$ *for* $B_{\mathsf{ek}} = B' + \frac{h+1}{2}$*,* $B_{\mathsf{rlk}} = B' + \frac{h^2+h+1}{2}$*,* $B_{\mathsf{rtk}} = B' + \frac{2h+1}{2}$*,* $B_{\mathsf{cjk}} = B' + \frac{2h+1}{2}$*,* $B_e = \mathtt{Estim}(\mathtt{CtoS}; h, B_{\mathsf{rlk}}, B_{\mathsf{rtk}}, B_{\mathsf{cjk}}, B' + \frac{h+3}{2}, \Delta)$*, and* $\mathcal{X} = \{\boldsymbol{x} \in R \mid \|\mathtt{Unpack}(\boldsymbol{x})\|_\infty \leq 1\}$.

[2] We do not specify the level consumption $\ell_{\mathtt{CtoS}}$ and error bound for the coeff-to-slot operation here, as it depends on which algorithm is used for implementation [BCH+24, CCS19, HS15, HS18].

$$\mathsf{R}'_{\mathsf{CKKS}} = \left\{ (\mathsf{x}'_{\mathsf{CKKS}}, \mathsf{w}'_{\mathsf{CKKS}}) \;\middle|\; \begin{array}{r} \|\boldsymbol{s}\|_\infty \le 1 \wedge \|\boldsymbol{s}\|_1 \le h \wedge \\ \left\|\boldsymbol{e}'_{\mathsf{ek}}\right\|_\infty, \left\|\vec{\boldsymbol{e}}'_{\mathsf{rlk}}\right\|_\infty, \left\|\vec{\boldsymbol{e}}'_{\mathsf{rtk}}\right\|_\infty, \left\|\vec{\boldsymbol{e}}'_{\mathsf{cjk}}\right\|_\infty, \left\|\vec{\boldsymbol{e}}'\right\|_\infty \le B' \wedge \\ \left\|\boldsymbol{m}'_0\right\|_\infty, \ldots, \left\|\boldsymbol{m}'_{k/2-1}\right\|_\infty \le \lfloor p/Q_L \cdot \Delta \rceil \wedge \\ \mathsf{ek}'_0 + \boldsymbol{s} \cdot \mathsf{ek}'_1 = \boldsymbol{e}'_{\mathsf{ek}} \wedge \\ \mathsf{rlk}'_0 + \boldsymbol{s} \cdot \mathsf{rlk}'_1 = \boldsymbol{s}^2 \cdot \lfloor p/Q_L \cdot \vec{g} \rceil + \vec{\boldsymbol{e}}'_{\mathsf{rtk}} \wedge \\ \mathsf{rtk}'_0 + \boldsymbol{s} \cdot \mathsf{rtk}'_1 = \varphi(\boldsymbol{s}) \cdot \lfloor p/Q_L \cdot \vec{g} \rceil + \vec{\boldsymbol{e}}'_{\mathsf{rtk}} \wedge \\ \mathsf{cjk}'_0 + \boldsymbol{s} \cdot \mathsf{cjk}'_1 = \psi(\boldsymbol{s}) \cdot \lfloor p/Q_L \cdot \vec{g} \rceil + \vec{\boldsymbol{e}}'_{\mathsf{cjk}} \wedge \\ \mathsf{ct}'_{i,0} + \boldsymbol{s} \cdot \mathsf{ct}'_{i,1} = \boldsymbol{m}'_i + \boldsymbol{e}'_i \text{ for } 0 \le i < k/2 \end{array} \right\}$$

Proof. See Appendix D.8. □

The above theorem states that it suffices to construct a ZKAoK for the relation $\mathsf{R}'_{\mathsf{CKKS}}$ in order for the sender to ensure that CKKS ciphertexts and public keys are well-formed. Below, we construct a ZKAoK for $\mathsf{R}'_{\mathsf{CKKS}}$ based on the PIOP framework.

Theorem 11. *An interactive protocol $\Pi_{\mathtt{CT\&PK}}$ described in Fig. 9 is an HVZK PIOP for a relation $\mathsf{R}'_{\mathsf{CKKS}}$ with a soundness error of $\frac{O(k+L+N)}{p-N}$, where T, P_φ, and P_ψ are matrices in $\mathbb{Z}_p^{N\times N}$ corresponding to $\mathtt{NTT}$, φ, and ψ, respectively, for the vector representation of coefficients.*

Proof. See Appendix D.7. □

$\Pi_{\mathtt{CT\&PK}}$

Instance: $\mathsf{pk}', \mathsf{ct}'_0, \ldots, \mathsf{ct}'_{k/2-1}$

Witness: $\boldsymbol{s}, \boldsymbol{e}'_{\mathsf{ek}} \in R_P$, $\vec{\boldsymbol{e}}'_{\mathsf{rlk}}, \vec{\boldsymbol{e}}'_{\mathsf{rtk}}, \vec{\boldsymbol{e}}'_{\mathsf{cjk}} \in R_P^L$, $\vec{\boldsymbol{e}}', \vec{\boldsymbol{m}}' \in R_P^{k/2}$

1. The prover $\mathtt{P}$ samples the followings.

$$\hat{\boldsymbol{s}} \leftarrow \mathtt{REcd}(\mathtt{Coeff}(\boldsymbol{s})),\ \underline{\hat{\boldsymbol{s}}} \leftarrow \mathtt{REcd}(\mathtt{NTT}(\boldsymbol{s}))$$
$$\underline{\hat{\boldsymbol{s}}}_{\mathsf{rtk}} \leftarrow \mathtt{REcd}(\mathtt{NTT}(\varphi(\boldsymbol{s}))),\ \underline{\hat{\boldsymbol{s}}}_{\mathsf{cjk}} \leftarrow \mathtt{REcd}(\mathtt{NTT}(\psi(\boldsymbol{s})))$$
$$\hat{\boldsymbol{e}}_{\mathsf{ek}} \leftarrow \mathtt{REcd}(\mathtt{Coeff}(\boldsymbol{e}'_{\mathsf{ek}})),\ \underline{\hat{\boldsymbol{e}}}_{\mathsf{ek}} \leftarrow \mathtt{REcd}(\mathtt{NTT}(\boldsymbol{e}'_{\mathsf{ek}}))$$
$$\hat{\boldsymbol{e}}_{\mathsf{rlk},j} \leftarrow \mathtt{REcd}(\mathtt{Coeff}(\boldsymbol{e}'_{\mathsf{rlk},j})),\ \underline{\hat{\boldsymbol{e}}}_{\mathsf{rlk},j} \leftarrow \mathtt{REcd}(\mathtt{NTT}(\boldsymbol{e}'_{\mathsf{rlk},j})) \text{ for } 0 \le j < L$$
$$\hat{\boldsymbol{e}}_{\mathsf{rtk},j} \leftarrow \mathtt{REcd}(\mathtt{Coeff}(\boldsymbol{e}'_{\mathsf{rtk},j})),\ \underline{\hat{\boldsymbol{e}}}_{\mathsf{rtk},j} \leftarrow \mathtt{REcd}(\mathtt{NTT}(\boldsymbol{e}'_{\mathsf{rtk},j})) \text{ for } 0 \le j < L$$
$$\hat{\boldsymbol{e}}_{\mathsf{cjk},j} \leftarrow \mathtt{REcd}(\mathtt{Coeff}(\boldsymbol{e}'_{\mathsf{cjk},j})),\ \underline{\hat{\boldsymbol{e}}}_{\mathsf{cjk},j} \leftarrow \mathtt{REcd}(\mathtt{NTT}(\boldsymbol{e}'_{\mathsf{cjk},j})) \text{ for } 0 \le j < L$$
$$\hat{\boldsymbol{e}}_i \leftarrow \mathtt{REcd}(\mathtt{Coeff}(\boldsymbol{e}'_i)),\ \underline{\hat{\boldsymbol{e}}}_i \leftarrow \mathtt{REcd}(\mathtt{NTT}(\boldsymbol{e}'_i)) \text{ for } 0 \le i < k/2$$
$$\hat{\boldsymbol{m}}_i \leftarrow \mathtt{REcd}(\mathtt{Coeff}(\boldsymbol{m}'_i)),\ \underline{\hat{\boldsymbol{m}}}_i \leftarrow \mathtt{REcd}(\mathtt{NTT}(\boldsymbol{m}'_i)) \text{ for } 0 \le i < k/2$$

   Then, the prover sends polynomial oracles $[\![\hat{\boldsymbol{s}}]\!]$, $[\![\underline{\hat{\boldsymbol{s}}}]\!]$, $[\![\underline{\hat{\boldsymbol{s}}}_{\mathsf{rtk}}]\!]$, $[\![\underline{\hat{\boldsymbol{s}}}_{\mathsf{cjk}}]\!]$, $[\![\hat{\boldsymbol{e}}_{\mathsf{ek}}]\!]$, $[\![\underline{\hat{\boldsymbol{e}}}_{\mathsf{ek}}]\!]$, $\{[\![\hat{\boldsymbol{e}}_{\mathsf{rlk},j}]\!], [\![\underline{\hat{\boldsymbol{e}}}_{\mathsf{rlk},j}]\!], [\![\hat{\boldsymbol{e}}_{\mathsf{rtk},j}]\!], [\![\underline{\hat{\boldsymbol{e}}}_{\mathsf{rtk},j}]\!], [\![\hat{\boldsymbol{e}}_{\mathsf{cjk},j}]\!], [\![\underline{\hat{\boldsymbol{e}}}_{\mathsf{cjk},j}]\!]\}_{0\le j<L}$, $\{[\![\hat{\boldsymbol{e}}_i]\!], [\![\underline{\hat{\boldsymbol{e}}}_i]\!], [\![\hat{\boldsymbol{m}}_i]\!], [\![\underline{\hat{\boldsymbol{m}}}_i]\!]\}_{0\le i<k/2}$ of degree $\le 2N-1$ to the verifier $\mathtt{V}$.
2. The verifier sends a random point $\delta \leftarrow \mathcal{U}(\mathbb{Z}_p \setminus \mathbb{H})$.
3. The prover $\mathtt{P}$ and the verifier $\mathtt{V}$ invoke the following PIOPs.

$$\Pi_{L^\infty}(1; [\![\hat{\boldsymbol{s}}]\!]),\ \Pi_{L^2}(h; [\![\hat{\boldsymbol{s}}]\!]),\ \Pi_{\mathtt{Lin}}(P_\varphi, P_\psi; [\![\underline{\hat{\boldsymbol{s}}}]\!], [\![\underline{\hat{\boldsymbol{s}}}]\!]; [\![\underline{\hat{\boldsymbol{s}}}_{\mathsf{rtk}}]\!], [\![\underline{\hat{\boldsymbol{s}}}_{\mathsf{cjk}}]\!]) \tag{1}$$

$$\Pi_{L^\infty}\left(\lfloor p/Q_L \cdot \Delta \rceil ; \{[\![\hat{\boldsymbol{m}}_i]\!]\}_{0\le i<k/2}\right) \tag{2}$$

$$\Pi_{L^\infty}\left(B'; [\![\hat{\boldsymbol{e}}_{\mathsf{ek}}]\!], \{[\![\hat{\boldsymbol{e}}_{\mathsf{rlk},j}]\!], [\![\hat{\boldsymbol{e}}_{\mathsf{rtk},j}]\!], [\![\hat{\boldsymbol{e}}_{\mathsf{cjk},j}]\!]\}_{0\le j<L}, \{[\![\hat{\boldsymbol{e}}_i]\!]\}_{0\le i<k/2}\right) \tag{3}$$

$$\Pi_{\mathtt{Lin}}\Big(T; [\![\hat{\boldsymbol{s}}]\!], [\![\hat{\boldsymbol{e}}_{\mathsf{ek}}]\!], \{[\![\hat{\boldsymbol{e}}_{\mathsf{rlk},j}]\!], [\![\hat{\boldsymbol{e}}_{\mathsf{rtk},j}]\!], [\![\hat{\boldsymbol{e}}_{\mathsf{cjk},j}]\!]\}_{0\le j<L}, \{[\![\hat{\boldsymbol{e}}_i]\!], [\![\hat{\boldsymbol{m}}_i]\!]\}_{0\le i<k/2} ; \tag{4}$$

$$[\![\underline{\hat{\boldsymbol{s}}}]\!], [\![\underline{\hat{\boldsymbol{e}}}_{\mathsf{ek}}]\!], \{[\![\underline{\hat{\boldsymbol{e}}}_{\mathsf{rlk},j}]\!], [\![\underline{\hat{\boldsymbol{e}}}_{\mathsf{rtk},j}]\!], [\![\underline{\hat{\boldsymbol{e}}}_{\mathsf{cjk},j}]\!]\}_{0\le j<L}, \{[\![\underline{\hat{\boldsymbol{e}}}_i]\!], [\![\underline{\hat{\boldsymbol{m}}}_i]\!]\}_{0\le i<k/2}\Big)$$

$$\Pi_{\mathtt{AC}}(\mathtt{NTT}(\mathsf{ek}'_0) + \mathtt{NTT}(\mathsf{ek}'_1) \odot \vec{X}_0 - \vec{X}_1; [\![\underline{\hat{\boldsymbol{s}}}]\!], [\![\underline{\hat{\boldsymbol{e}}}_{\mathsf{ek}}]\!]) \tag{5}$$

$$\Pi_{\mathtt{AC}}\Big(\sum_{j=0}^{L-1} \delta^j \cdot \Big(\mathtt{NTT}(\mathsf{rlk}'_{0,j}) + \mathtt{NTT}(\mathsf{rlk}'_{1,j}) \odot \vec{X}_0 - \lfloor p/q_i \rceil \cdot \vec{X}_0^2 - \vec{X}_{j+1}\Big); \tag{6}$$

$$[\![\underline{\hat{\boldsymbol{s}}}]\!], \{[\![\underline{\hat{\boldsymbol{e}}}_{\mathsf{rlk},j}]\!]\}_{0\le j<L}\Big)$$

$$\Pi_{\mathtt{AC}}\Big(\sum_{j=0}^{L-1} \delta^j \cdot \Big(\mathtt{NTT}(\mathsf{rtk}'_{0,j}) + \mathtt{NTT}(\mathsf{rtk}'_{1,j}) \odot \vec{X}_0 - \lfloor p/q_i \rceil \cdot \vec{X}_1 - \vec{X}_{j+2}\Big); \tag{7}$$

$$[\![\underline{\hat{\boldsymbol{s}}}]\!], [\![\underline{\hat{\boldsymbol{s}}}_{\mathsf{rtk}}]\!], \{[\![\underline{\hat{\boldsymbol{e}}}_{\mathsf{rtk},j}]\!]\}_{0\le j<L}\Big)$$

$$\Pi_{\mathtt{AC}}\Big(\sum_{j=0}^{L-1} \delta^j \cdot \Big(\mathtt{NTT}(\mathsf{cjk}'_{0,j}) + \mathtt{NTT}(\mathsf{cjk}'_{1,j}) \odot \vec{X}_0 - \lfloor p/q_i \rceil \cdot \vec{X}_1 - \vec{X}_{j+2}\Big); \tag{8}$$

$$[\![\underline{\hat{\boldsymbol{s}}}]\!], [\![\underline{\hat{\boldsymbol{s}}}_{\mathsf{cjk}}]\!], \{[\![\underline{\hat{\boldsymbol{e}}}_{\mathsf{cjk},j}]\!]\}_{0\le j<L}\Big)$$

$$\Pi_{\mathtt{AC}}\Big(\sum_{i=0}^{k/2-1} \delta^i \cdot \Big(\mathtt{NTT}(\mathsf{ct}'_{0,i}) + \mathtt{NTT}(\mathsf{ct}'_{1,i}) \odot \vec{X}_0 - \vec{X}_{2i+1} - \vec{X}_{2i+2}\Big); \tag{9}$$

$$[\![\underline{\hat{\boldsymbol{s}}}]\!], \{[\![\underline{\hat{\boldsymbol{e}}}_i]\!], [\![\underline{\hat{\boldsymbol{m}}}_i]\!]\}_{0\le i<k/2}\Big)$$

Fig. 9: PIOP for CKKS ciphertexts and public key

6 Evaluation

In this section, we present concrete parameters and benchmark results for our ZKAoK for CKKS. To compile PIOPs into non-interactive arguments, we use the HSS scheme [HSS24] as the underlying polynomial commitment scheme (PCS), as it supports large, NTT-friendly prime fields and offers fast proof generation speeds.

6.1 Parameter Setting

For the CKKS scheme parameters, we set $N = 2^{14}$ and $\log Q_L = 404$, with $L + 1 = 9$ distinct primes, where $q_0, \ldots, q_{L-1}$ are each 43 bits in length, and q_L is 60 bits. For the secret key and error distributions, we use a sparse ternary

distribution with Hamming weight 256 and a uniform ternary distribution, respectively. The selected parameters provide 128 bits of security, as verified using the lattice estimator [APS15].

For the HSS scheme parameters, we select values that minimize the proof size. For the base modulus p, we choose an NTT-friendly prime of similar size to q. In the HSS scheme, the modulus p is of the form $b^r + 1$. To ensure that p is NTT-friendly, it suffices to set b as an even number and $r \geq \log N$. Therefore, we set $b = 10792$ and $r = 32$, so that $p \approx q$. For the remaining parameters of the HSS scheme, we use $\lfloor \log \mathsf{q} \rceil = 100$, $\mathsf{n} = 2^{13}$, $\mathsf{d} = 2^{11}$, $\mu = 1$, and $\nu = 2$. For a detailed explanation of these parameters, we refer to the parameter setting section in [HSS24].

6.2 Benchmark Results

We implement $\Pi_{\mathtt{CT\&PK}}$ and present the benchmark results. Our implementation is written in Rust, building on the implementation of the HSS scheme available at `https://github.com/SNUCP/celpc`. Our source code is available at `https://github.com/SNUCP/ckks-piop`. All experiments were conducted on a machine equipped with an Intel(R) Xeon(R) Platinum 8268 CPU running at 2.90GHz, using a single thread. We measure performance by varying the number of ciphertexts $k = 2, 4, 8$ and scaling factor $\log \Delta = 16, 32$. The results are summarized in Table 1.

k	$\log \Delta$	PK Size	CT Size	Proof Size	Prover Time	Verifier Time
2	16	39.5 MB	1.57 MB	17.9 MB	324.35s	50.88s
4	16		3.14 MB	18.9 MB	365.46s	56.08s
8	16		6.28 MB	21.0 MB	442.86s	67.13s
2	32		1.57 MB	18.7 MB	356.90s	54.65s
4	32		3.14 MB	20.4 MB	425.26s	64.33s
8	32		6.28 MB	24.0 MB	561.28s	83.63s

Table 1: Proof sizes and benchmark results of $\Pi_{\mathtt{CT\&PK}}$.

We note that our proof size is smaller than the total size of the CKKS ciphertext and public keys, as the HSS scheme results in a proof size that is proportional to the square root of the witness size. The performance overhead with respect to the number of ciphertexts is relatively moderate, since a large portion of the proof generation process is dedicated to handling public keys, which are significantly larger than ciphertexts.

Since our ZKAoK is the first construction for the CKKS scheme, we could not report direct performance comparisons with other works. However, we note that the computational overhead introduced by our ZKAoK is comparable to that of

existing ZKAoKs for homomorphic encryption [CMS+23,HLSS25]. Furthermore, this overhead is considered affordable in CKKS-based PPML applications, which are typically computationally intensive. For example, in [LLL+22], a single CNN inference takes approximately one hour of CPU time. Thus, adding a few minutes for the validation of CKKS public keys and ciphertexts does not significantly affect the overall performance in such scenarios.

7 Conclusion

In this work, we analyze the security of a CKKS-based homomorphic evaluation protocol in a sender-receiver setting. To address challenges in sender privacy analysis, we introduce a new concept: the differentially private homomorphic evaluation (DPHE) protocol, whose security is analyzed within the framework of differential privacy. To achieve the DPHE property, we propose a general compilation method based on the Laplace mechanism. In addition, to handle malicious receivers, we construct a ZKAoK for CKKS based on the PIOP framework. Finally, to demonstrate the practicality of our approach, we report benchmark results for the ZKAoK for CKKS, obtained through PIOP compilation using the HSS polynomial commitment scheme [HSS24].

We expect that our approach can be extended to multi-party variants of CKKS [KKL+23,KLSW24,CDKS19], where achieving indistinguishability-based security is also challenging for similar reasons as in CKKS-based homomorphic evaluation protocols. Adopting our differential privacy-based analysis would result in a more robust security analysis in such scenarios.

Acknowledgement

This work was partly supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2023-00211649) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. RS-2024-00399491, Development of Privacy-Preserving Multiparty Computation Techniques for Secure Multiparty Data Integration).

References

ACG+16. Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

ACLS18. Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. Pir with compressed queries and amortized query processing. In *2018 IEEE symposium on security and privacy (SP)*, pages 962–979. IEEE, 2018.

ADDG24. Martin R Albrecht, Alex Davidson, Amit Deo, and Daniel Gardham. Crypto dark matter on the torus: Oblivious prfs from shallow prfs and tfhe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 447–476. Springer, 2024.

AJLA+12. Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 483–501. Springer, 2012.

ALS20. Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. Practical product proofs for lattice commitments. In *Annual International Cryptology Conference*, pages 470–499. Springer, 2020.

APS15. Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

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

BCCW19. Fabian Boemer, Anamaria Costache, Rosario Cammarota, and Casimir Wierzynski. ngraph-he2: A high-throughput framework for neural network inference on encrypted data. In *Proceedings of the 7th ACM workshop on encrypted computing & applied homomorphic cryptography*, pages 45–56, 2019.

BCFK21. Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. Flexible and efficient verifiable computation on encrypted data. In *IACR International Conference on Public-Key Cryptography*, pages 528–558. Springer, 2021.

BCH+24. Youngjin Bae, Jung Hee Cheon, Guillaume Hanrot, Jai Hyun Park, and Damien Stehlé. Plaintext-ciphertext matrix multiplication and fhe bootstrapping: Fast and fused. In *Annual International Cryptology Conference*, pages 387–421. Springer, 2024.

BCOS20. Cecilia Boschini, Jan Camenisch, Max Ovsiankin, and Nicholas Spooner. Efficient post-quantum snarks for rsis and rlwe and their applications to privacy. In *Post-Quantum Cryptography: 11th International Conference, PQCrypto 2020, Paris, France, April 15–17, 2020, Proceedings 11*, pages 247–267. Springer, 2020.

BGL+23. James Bell, Adrià Gascón, Tancrède Lepoint, Baiyu Li, Sarah Meiklejohn, Mariana Raykova, and Cathie Yun. {ACORN}: input validation for secure aggregation. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4805–4822, 2023.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325, 2012.

BKSS24. Youngjin Bae, Jaehyung Kim, Damien Stehlé, and Elias Suvanto. Bootstrapping small integers with ckks. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 330–360, 2024.

BNO08. Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *Advances in Cryptology–CRYPTO 2008: 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings 28*, pages 451–468. Springer, 2008.

Bra12. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual cryptology conference*, pages 868–886. Springer, 2012.

BSCR+19. Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P Ward. Aurora: Transparent succinct arguments for r1cs. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 103–128. Springer, 2019.

BSCS16. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Proceedings, Part II, of the 14th International Conference on Theory of Cryptography-Volume 9986*, pages 31–60, 2016.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.

CCP+24. Jung Hee Cheon, Hyeongmin Choe, Alain Passelègue, Damien Stehlé, and Elias Suvanto. Attacks against the ind-cpad security of exact fhe schemes. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 2505–2519, 2024.

CCS19. Hao Chen, Ilaria Chillotti, and Yongsoo Song. Improved bootstrapping for approximate homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 34–54. Springer, 2019.

CDKS19. Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient multikey homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 395–412, 2019.

CGGI20. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

CHK+18a. Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. *Advances in Cryptology–EUROCRYPT 2018*, pages 360–384, 2018.

CHK+18b. Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full rns variant of approximate homomorphic encryption. In *International Conference on Selected Areas in Cryptography*, pages 347–368, 2018.

CHK+24. Wonhee Cho, Guillaume Hanrot, Taeseong Kim, Minje Park, and Damien Stehlé. Fast and accurate homomorphic softmax evaluation. In *ACM CCS*, 2024.

CHLR18. Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled psi from fully homomorphic encryption with malicious security. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1223–1237, 2018.

CHM+20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zksnarks with universal and updatable srs. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 738–768, 2020.

CJM20. Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. Cryptanalytic extraction of neural network models. In *Annual international cryptology conference*, pages 189–218. Springer, 2020.

CKK20. Jung Hee Cheon, Dongwoo Kim, and Duhyeong Kim. Efficient homomorphic comparison methods with optimal complexity. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26*, pages 221–256. Springer, 2020.

CKKS17. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.

CLE+19. Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX security symposium (USENIX security 19)*, pages 267–284, 2019.

CMdG+21. Kelong Cong, Radames Cruz Moreno, Mariana Botelho da Gama, Wei Dai, Ilia Iliashenko, Kim Laine, and Michael Rosenberg. Labeled psi from homomorphic encryption with reduced computation and communication. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1135–1150, 2021.

CMS+23. Sylvain Chatel, Christian Mouchet, Ali Utkan Sahin, Apostolos Pyrgelis, Carmela Troncoso, and Jean-Pierre Hubaux. Pelta-shielding multiparty-fhe against malicious adversaries. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 711–725, 2023.

CSBB24. Marina Checri, Renaud Sirdey, Aymen Boudguiga, and Jean-Paul Bultel. On the practical cpa d security of "exact" and threshold fhe schemes and libraries. In *Annual International Cryptology Conference*, pages 3–33. Springer, 2024.

CTW+21. Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

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

ceedings of the 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pages 35–46, 2023.

DF18. Cynthia Dwork and Vitaly Feldman. Privacy-preserving prediction. In *Conference On Learning Theory*, pages 1693–1702. PMLR, 2018.

DMNS06. Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. *Theory of Cryptography*, pages 265–284, 2006.

ENS20. Muhammed F Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 259–288. Springer, 2020.

FIM+01. Joan Feigenbaum, Yuval Ishai, Tal Malkin, Kobbi Nissim, Martin J Strauss, and Rebecca N Wright. Secure multiparty computation of approximations. In *Automata, Languages and Programming: 28th International Colloquium, ICALP 2001 Crete, Greece, July 8–12, 2001 Proceedings 28*, pages 927–938. Springer, 2001.

FNP20. Dario Fiore, Anca Nitulescu, and David Pointcheval. Boosting verifiable computation on encrypted data. In *Public-Key Cryptography–PKC 2020: 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4–7, 2020, Proceedings, Part II 23*, pages 124–154. Springer, 2020.

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

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 169–178. ACM, 2009.

GNSJ24. Qian Guo, Denis Nabokov, Elias Suvanto, and Thomas Johansson. Key recovery attacks on approximate homomorphic encryption with non-worst-case noise flooding countermeasures. In *Usenix Security*, 2024.

GNSV23. Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: Snarks for ring arithmetic. *Journal of Cryptology*, 36(4):41, 2023.

HLSS25. Intak Hwang, Hyeonbum Lee, Jinyeong Seo, and Yongsoo Song. Practical zero-knowledge PIOP for maliciously secure multiparty homomorphic encryption. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2025.

HS15. Shai Halevi and Victor Shoup. Bootstrapping for helib. In *Annual International conference on the theory and applications of cryptographic techniques*, pages 641–670. Springer, 2015.

HS18. Shai Halevi and Victor Shoup. Faster homomorphic linear transformations in HElib. In *Annual International Cryptology Conference*, pages 93–120. Springer, 2018.

HSS24. Intak Hwang, Jinyeong Seo, and Yongsoo Song. Concretely efficient lattice-based polynomial commitment from standard assumptions. In *Annual International Cryptology Conference*, pages 414–448. Springer, 2024.

JPK+24. Jae Hyung Ju, Jaiyoung Park, Jongmin Kim, Minsik Kang, Donghwan Kim, Jung Hee Cheon, and Jung Ho Ahn. Neujeans: Private neural network inference with joint optimization of convolution and fhe bootstrapping. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 4361–4375, 2024.

KKL+23. Taechan Kim, Hyesun Kwak, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Asymptotically faster multi-key homomorphic encryption from homomorphic gadget decomposition. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 726–740, 2023.

KLSS23. Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Toward practical lattice-based proof of knowledge from hint-mlwe. In *Annual International Cryptology Conference*, pages 549–580. Springer, 2023.

KLSW24. Hyesun Kwak, Dongwon Lee, Yongsoo Song, and Sameer Wagh. A general framework of homomorphic encryption for multiple parties with non-interactive key-aggregation. In *International Conference on Applied Cryptography and Network Security*, pages 403–430. Springer, 2024.

LAG+19. Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE symposium on security and privacy (SP)*, pages 656–672. IEEE, 2019.

LATV12. Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234, 2012.

LLL+21. Joon-Woo Lee, Eunsang Lee, Yongwoo Lee, Young-Sik Kim, and Jong-Seon No. High-precision bootstrapping of rns-ckks homomorphic encryption using optimal minimax polynomial approximation and inverse sine function. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 618–647, 2021.

LLL+22. Eunsang Lee, Joon-Woo Lee, Junghyun Lee, Young-Sik Kim, Yongjune Kim, Jong-Seon No, and Woosuk Choi. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In *International Conference on Machine Learning*, pages 12403–12422. PMLR, 2022.

LM21. Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 648–677. Springer, 2021.

LMSWS22. Baiyu Li, Daniele Micciancio, Mark Schultz-Wu, and Jessica Sorrell. Securing approximate homomorphic encryption using differential privacy. In *Annual International Cryptology Conference*, pages 560–589. Springer, 2022.

LNS20. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Practical lattice-based zero-knowledge proofs for integer relations. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 1051–1070, 2020.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pages 1–23. Springer, 2010.

MPRV09. Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational differential privacy. In *Annual International Cryptology Conference*, pages 126–142. Springer, 2009.

MR07. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

MV24. Daniele Micciancio and Vinod Vaikuntanathan. Sok: Learning with errors, circular security, and fully homomorphic encryption. In *IACR International Conference on Public-Key Cryptography*, pages 291–321. Springer, 2024.

MW22. Samir Jordan Menon and David J Wu. Spiral: Fast, high-rate single-server pir via fhe composition. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 930–947. IEEE, 2022.

NP06. Moni Naor and Benny Pinkas. Oblivious polynomial evaluation. *SIAM Journal on Computing*, 35(5):1254–1281, 2006.

Reg09. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

SSSS17. Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

SSSSS10. Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability, stability and uniform convergence. *The Journal of Machine Learning Research*, 11:2635–2670, 2010.

A CKKS Error Analysis

This section provides an error analysis for each homomorphic operation. In the analysis below, we assume that our chosen set of prime modulus $Q_L = q_1 \dots q_L$ satisfies the condition $q_L > q_i LN$ for all q_i $(1 \le i < L)$. It is important to note that selecting q_L in this manner is a common practice in practical parameter settings, as it helps reduce noise introduction during homomorphic computations. Additionally, for simplicity, we will use the same notation as in the algorithm description within the proof.

Lemma 3 (External Product). *Suppose that* $\vec{\boldsymbol{u}}_0 + \vec{\boldsymbol{u}}_1 \boldsymbol{s} = \boldsymbol{\mu} \cdot \vec{g} + \vec{\boldsymbol{e}} \pmod{Q_L}$ *for some vectors* $\vec{\boldsymbol{u}}_0 = (\boldsymbol{u}_{0,0}, \dots, \boldsymbol{u}_{0,L}), \vec{\boldsymbol{u}}_1 = (\boldsymbol{u}_{1,0}, \dots, \boldsymbol{u}_{1,L}) \in R_{Q_L}^L$ *and ring element* $\boldsymbol{\mu} \in R_{Q_{L-1}}$. *Then, the following equation holds for any* $\boldsymbol{a} \in R_{Q_\ell}$ *and* $\vec{\boldsymbol{e}} = (\boldsymbol{e}_0, \dots, \boldsymbol{e}_L) \in R_{Q_L}^L$ *where* $0 \le \ell < L$.

$$\mathtt{Err}_{\mathsf{sk}}\left(\boldsymbol{a} \boxdot (\vec{\boldsymbol{u}}_0, \vec{\boldsymbol{u}}_1), \boldsymbol{\mu} \cdot \boldsymbol{a}\right) \le \frac{\|\vec{\boldsymbol{e}}\|_\infty + h + 1}{2}.$$

Proof. By definition, we have

$$\begin{aligned}
&\mathtt{Dec}_{\mathsf{sk}}(\boldsymbol{a} \boxdot (\vec{\boldsymbol{u}}_0, \vec{\boldsymbol{u}}_1)) = \boldsymbol{a} \boxdot \vec{\boldsymbol{u}}_0 + (\boldsymbol{a} \boxdot \vec{\boldsymbol{u}}_1) \cdot \boldsymbol{s} \\
&= \frac{1}{q_L} \sum_{i=0}^{\ell-1} [(Q_{L-1}/q_i)^{-1} \cdot \boldsymbol{a}]_{q_i} (\boldsymbol{u}_{0,i} + \boldsymbol{u}_{1,i} \cdot \boldsymbol{s}) - \boldsymbol{r}_0 - \boldsymbol{r}_1 \cdot \boldsymbol{s} \pmod{Q_\ell}
\end{aligned}$$

where

$$\boldsymbol{r}_i = \frac{1}{q_L} \sum_{j=0}^{\ell-1} \left[(Q_{L-1}/q_j)^{-1} \cdot \boldsymbol{a}\right]_{q_j} \boldsymbol{u}_{i,j} - \left\lfloor \frac{1}{q_L} \sum_{j=0}^{\ell-1} \left[(Q_{L-1}/q_j)^{-1} \cdot \boldsymbol{a}\right]_{q_j} \boldsymbol{u}_{i,j} \right\rceil$$

for $i = 0, 1$.

Note that $\boldsymbol{u}_{0,i} + \boldsymbol{u}_{1,i} \cdot \boldsymbol{s} = \boldsymbol{\mu} \cdot Q_L/q_i + \boldsymbol{e}_i \pmod{Q_\ell \cdot q_L}$ since $Q_\ell \cdot q_L \mid Q_L$, we have the following by definition.

$$\begin{aligned}
&\sum_{i=0}^{\ell-1} [(Q_{L-1}/q_i)^{-1} \cdot \boldsymbol{a}]_{q_i} (\boldsymbol{u}_{0,i} + \boldsymbol{u}_{1,i} \cdot \boldsymbol{s}) \\
&= \sum_{i=0}^{\ell-1} [(Q_{L-1}/q_i)^{-1} \cdot \boldsymbol{a}]_{q_i} (\boldsymbol{\mu} \cdot Q_L/q_i + \boldsymbol{e}_i) \\
&= q_L \cdot \boldsymbol{\mu} \cdot \boldsymbol{a} + \sum_{i=0}^{\ell-1} [(Q_{L-1}/q_i)^{-1} \cdot \boldsymbol{a}]_{q_i} \cdot \boldsymbol{e}_i \pmod{Q_\ell \cdot q_L}
\end{aligned}$$

Note that the final equality comes from the Chinese Remainder Theorem. (Consider modulo q_i for $i = 0, \dots, \ell$ and L.) Therefore, we can conclude the following inequality:

$\mathtt{Err}_{\mathsf{sk}}(\boldsymbol{a} \boxdot (\vec{\boldsymbol{u}}_0, \vec{\boldsymbol{u}}_1), \boldsymbol{\mu} \cdot \boldsymbol{a})$

$$
\begin{aligned}
&= \left\| \frac{1}{q_L} \sum_{i=0}^{\ell-1} [(Q_{L-1}/q_i)^{-1} \cdot \boldsymbol{a}]_{q_i} (\boldsymbol{u}_{0,i} + \boldsymbol{u}_{1,i} \cdot \boldsymbol{s}) - \boldsymbol{r}_0 - \boldsymbol{r}_1 \cdot \boldsymbol{s} - \boldsymbol{\mu} \cdot \boldsymbol{a} \pmod{Q_\ell} \right\|_\infty \\
&= \left\| \frac{1}{q_L} \sum_{i=0}^{\ell-1} [(Q_{L-1}/q_i)^{-1} \cdot \boldsymbol{a}]_{q_i} \cdot \boldsymbol{e}_i - \boldsymbol{r}_0 - \boldsymbol{r}_1 \cdot \boldsymbol{s} \right\|_\infty \\
&\leq \frac{1}{q_L} \left\| \sum_{i=0}^{\ell-1} [(Q_{L-1}/q_i)^{-1} \cdot \boldsymbol{a}]_{q_i} \cdot \boldsymbol{e}_i \right\|_\infty + \|\boldsymbol{r}_0 + \boldsymbol{r}_1 \cdot \boldsymbol{s}\|_\infty
\end{aligned}
$$

where the second equality is derived from the smallness of the term which does not incur the wraparound. Now, since $\left\| [(Q_{L-1}/q_i)^{-1} \cdot \boldsymbol{a}]_{q_i} \right\|_\infty \leq \frac{q_i}{2} < \frac{q_L}{2LN}$ by the assumption, the first term is bounded by

$$
\frac{1}{q_L} \left\| \sum_{i=0}^{\ell-1} [(Q_{L-1}/q_i)^{-1} \cdot \boldsymbol{a}]_{q_i} \cdot \boldsymbol{e}_i \right\|_\infty < \frac{1}{q_L} \cdot \ell \cdot \frac{q_L}{2LN} \cdot N \|\vec{\boldsymbol{e}}\|_\infty \leq \frac{\|\vec{\boldsymbol{e}}\|_\infty}{2}.
$$

On the other hand, since $\|\boldsymbol{r}_i\|_\infty \leq 1/2$, it is easy to show that $\|\boldsymbol{r}_0 + \boldsymbol{r}_1 \cdot \boldsymbol{s}\|_\infty \leq \frac{h+1}{2}$ from the fact that $\boldsymbol{s}$ is a ternary polynomial with Hamming weight at most h. Hence, we can obtain the desired noise bound $(\|\vec{\boldsymbol{e}}\|_\infty + h + 1)/2$. □

Lemma 4 (Homomorphic Addition). *Given CKKS ciphertexts* $\mathsf{ct}, \mathsf{ct}' \in R_{Q_\ell}^2$ *for* $0 \leq \ell < L$*, let* $\mathsf{ct}_{add} \leftarrow \underline{\mathtt{Add}}(\mathsf{ct}, \mathsf{ct}')$*. Then, it satisfies that*

$$
\mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}_{add}, \boldsymbol{m} + \boldsymbol{m}') \leq \mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}, \boldsymbol{m}) + \mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}', \boldsymbol{m}').
$$

Proof. Note that $\mathsf{ct}_{add} = (\boldsymbol{c}_0 + \boldsymbol{c}_0', \boldsymbol{c}_1 + \boldsymbol{c}_1') \in R_{Q_\ell}$. By definition, we obtain

$$
\begin{aligned}
\mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}_{add}, \boldsymbol{m} + \boldsymbol{m}') &= \|\mathtt{Dec}_{\mathsf{sk}}(\mathsf{ct}_{add}) - \boldsymbol{m} - \boldsymbol{m}'\|_\infty \\
&= \|[\boldsymbol{c}_0 + \boldsymbol{c}_0' + (\boldsymbol{c}_1 + \boldsymbol{c}_1')\boldsymbol{s}]_{Q_\ell} - \boldsymbol{m} - \boldsymbol{m}'\|_\infty \\
&\leq \|[\boldsymbol{c}_0 + \boldsymbol{c}_1 \boldsymbol{s} - \boldsymbol{m}]_{Q_\ell}\|_\infty + \|[\boldsymbol{c}_0' + \boldsymbol{c}_1' \boldsymbol{s} - \boldsymbol{m}']_{Q_\ell}\|_\infty \\
&= \mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}, \boldsymbol{m}) + \mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}', \boldsymbol{m}'),
\end{aligned}
$$

where the inequality is derived from the smallness of the errors and the triangle inequality. □

Lemma 5 (Homomorphic Multiplication). *Given CKKS ciphertexts* $\mathsf{ct}, \mathsf{ct}' \in R_{Q_\ell}^2$ *for* $0 \leq \ell < L$*, let* $\mathsf{ct}_{mul} \leftarrow \underline{\mathtt{Mul}_{\mathsf{rlk}}}(\mathsf{ct}, \mathsf{ct}')$*. Then, it satisfies that*

$$
\mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}_{mul}, \boldsymbol{m}\boldsymbol{m}') \leq N\|\boldsymbol{m}'\|_\infty E + N\|\boldsymbol{m}\|_\infty E' + NEE' + \frac{B_{\mathsf{rlk}} + h + 1}{2}
$$

for $E = \mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}, \boldsymbol{m})$*,* $E' = \mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}', \boldsymbol{m}')$ *and* B_{rlk} *is the infinity-norm bound of the relinearization key error.*

Proof. Let us parse $\boldsymbol{d}_2 \boxdot \mathsf{rlk} = (\boldsymbol{f}_0, \boldsymbol{f}_1) \in R_{Q_\ell}$. Then, by Lemma 3, we have

$$
\left\| \left[\boldsymbol{f}_0 + \boldsymbol{f}_1 \boldsymbol{s} - \boldsymbol{d}_2 \cdot \boldsymbol{s}^2 \right]_{Q_\ell} \right\|_\infty = \left\| \left[\boldsymbol{f}_0 + \boldsymbol{f}_1 \boldsymbol{s} - \boldsymbol{c}_1 \boldsymbol{c}_1' \cdot \boldsymbol{s}^2 \right]_{Q_\ell} \right\|_\infty \leq \frac{B_{\mathsf{rlk}} + h + 1}{2}.
$$

Next, let $\boldsymbol{e}$ and $\boldsymbol{e}'$ be the 'noise' of the ciphertexts ct and ct' which encrypt $\boldsymbol{m}$ and $\boldsymbol{m}'$, respectively. In other words, $\underline{\mathtt{Dec}_{\mathsf{sk}}}(\mathsf{ct}) = \boldsymbol{m} + \boldsymbol{e}$ and $\underline{\mathtt{Dec}_{\mathsf{sk}}}(\mathsf{ct}') = \boldsymbol{m}' + \boldsymbol{e}'$. Then, we can deduce that

$$
\begin{aligned}
&\mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}_{mul}, \boldsymbol{m}\boldsymbol{m}') \\
&= \left\| [\boldsymbol{c}_0\boldsymbol{c}_0' + (\boldsymbol{c}_0\boldsymbol{c}_1' + \boldsymbol{c}_0'\boldsymbol{c}_1)\boldsymbol{s} + (\boldsymbol{f}_0 + \boldsymbol{f}_1\boldsymbol{s})]_{Q_\ell} - \boldsymbol{m}\boldsymbol{m}' \right\|_\infty \\
&\le \left\| \left[\boldsymbol{c}_0\boldsymbol{c}_0' + (\boldsymbol{c}_0\boldsymbol{c}_1' + \boldsymbol{c}_0'\boldsymbol{c}_1)\boldsymbol{s} + \boldsymbol{c}_1\boldsymbol{c}_1'\boldsymbol{s}^2\right]_{Q_\ell} - \boldsymbol{m}\boldsymbol{m}' \right\|_\infty + \left\| \left[\boldsymbol{f}_0 + \boldsymbol{f}_1\boldsymbol{s} - \boldsymbol{c}_1\boldsymbol{c}_1' \cdot \boldsymbol{s}^2\right]_{Q_\ell} \right\|_\infty \\
&\le \left\| \left[\boldsymbol{c}_0\boldsymbol{c}_0' + (\boldsymbol{c}_0\boldsymbol{c}_1' + \boldsymbol{c}_0'\boldsymbol{c}_1)\boldsymbol{s} + \boldsymbol{c}_1\boldsymbol{c}_1'\boldsymbol{s}^2\right]_{Q_\ell} - \boldsymbol{m}\boldsymbol{m}' \right\|_\infty + \frac{B_{\mathsf{rlk}} + h + 1}{2} \\
&= \left\| (\boldsymbol{m} + \boldsymbol{e})(\boldsymbol{m}' + \boldsymbol{e}') - \boldsymbol{m}\boldsymbol{m}' \right\|_\infty + \frac{B_{\mathsf{rlk}} + h + 1}{2} \\
&\le N\|\boldsymbol{m}\|_\infty\|\boldsymbol{e}'\|_\infty + N\|\boldsymbol{m}'\|_\infty\|\boldsymbol{e}\|_\infty + N\|\boldsymbol{e}\|_\infty\|\boldsymbol{e}'\|_\infty + \frac{B_{\mathsf{rlk}} + h + 1}{2}.
\end{aligned}
$$

Since $\|\boldsymbol{e}\|_\infty \le \mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}, \boldsymbol{m})$ and $\|\boldsymbol{e}'\|_\infty \le \mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}', \boldsymbol{m}')$ by definition, we proved our claim. □

Lemma 6 (Homomorphic Rotation). *Given CKKS ciphertext $\mathsf{ct} \in R_{Q_\ell}^2$ for $0 \le \ell < L$, let $\mathsf{ct}' \leftarrow \underline{\mathtt{Rot}_{\mathsf{rtk}}}(\mathsf{ct})$. Then, it satisfies that*

$$\mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}', \varphi(\boldsymbol{m})) \le \mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}, \boldsymbol{m}) + \frac{B_{\mathsf{rtk}} + h + 1}{2}$$

where B_{rtk} is the infinity norm bound of the rotation key error.

Proof. By Lemma 3, we have

$$
\begin{aligned}
&\mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}', \varphi(\boldsymbol{m})) \\
&= \left\| [\varphi(\boldsymbol{c}_0) + \varphi(\boldsymbol{c}_1) \boxdot \mathsf{rtk}]_{Q_\ell} - \varphi(\boldsymbol{m}) \right\|_\infty \\
&\le \left\| [\varphi(\boldsymbol{c}_0) + \varphi(\boldsymbol{c}_1) \cdot \varphi(\boldsymbol{s})]_{Q_\ell} - \varphi(\boldsymbol{m}) \right\|_\infty + \left\| [\varphi(\boldsymbol{c}_1) \boxdot \mathsf{rtk} - \varphi(\boldsymbol{c}_1) \cdot \varphi(\boldsymbol{s})]_{Q_\ell} \right\|_\infty \\
&\le \left\| [\varphi(\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} - \boldsymbol{m})]_{Q_\ell} \right\|_\infty + \frac{B_{\mathsf{rtk}} + h + 1}{2} = \mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}, \boldsymbol{m}) + \frac{B_{\mathsf{rtk}} + h + 1}{2}
\end{aligned}
$$

where the last equality is derived from the fact that the automorphism $X \mapsto X^5$ only rearranges the coefficients up to sign. □

Lemma 7 (Homomorphic Conjugation). *Given CKKS ciphertext $\mathsf{ct} \in R_{Q_\ell}^2$ for $0 \le \ell < L$, let $\mathsf{ct}' \leftarrow \underline{\mathtt{Conj}_{\mathsf{cjk}}}(\mathsf{ct})$. Then it satisfies that*

$$\mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}', \psi(\boldsymbol{m})) \le \mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}, \boldsymbol{m}) + \frac{B_{\mathsf{cjk}} + h + 1}{2}$$

where B_{cjk} is the infinity norm bound of the rotation key.

Proof. The proof is essentially identical to the proof of Lemma 6. □

Lemma 8 (Rounding Operation). *Given CKKS ciphertext* $\mathsf{ct} \in R^2_{Q_\ell}$ *for* $0 \le \ell < L$*, let* $\mathsf{ct}' \leftarrow \underline{\mathtt{Round}}(\mathsf{ct})$*. Then, it satisfies that*

$$\mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}', \boldsymbol{m}/q_\ell) \le \frac{h+1}{2}.$$

Proof. Let $\underline{\mathtt{Dec}_{\mathsf{sk}}}(\mathsf{ct}) = \boldsymbol{m} + \boldsymbol{e}$ for some small noise $\boldsymbol{e}$. Then,

$$\begin{aligned}[\lfloor \boldsymbol{c}_0/q_\ell \rceil + \lfloor \boldsymbol{c}_1/q_\ell \rceil \cdot \boldsymbol{s}]_{Q_{\ell-1}} &= \frac{1}{q_\ell}\left[\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}\right]_{Q_\ell} - \boldsymbol{r}_0 - \boldsymbol{r}_1\boldsymbol{s} \\ &= \frac{1}{q_\ell}(\boldsymbol{m} + \boldsymbol{e}) - \boldsymbol{r}_0 - \boldsymbol{r}_1\boldsymbol{s}\end{aligned}$$

where

$$\boldsymbol{r}_i = \frac{1}{q_\ell}\boldsymbol{c}_i - \left\lfloor \frac{1}{q_\ell}\boldsymbol{c}_i \right\rceil$$

for $i = 0, 1$. Analogous to the proof of Lemma 3, we can deduce that $\|\boldsymbol{r}_i\|_\infty \le 1/2$. Therefore, from the definition,

$$\begin{aligned}\mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}', \boldsymbol{m}/q_\ell) &= \left\| [\lfloor \boldsymbol{c}_0/q_\ell \rceil + \lfloor \boldsymbol{c}_1/q_\ell \rceil \cdot \boldsymbol{s}]_{Q_{\ell-1}} - \boldsymbol{m}/q_\ell \right\|_\infty \\ &= \left\| \frac{1}{q_\ell}\boldsymbol{e} - \boldsymbol{r}_0 - \boldsymbol{r}_1\boldsymbol{s} \right\|_\infty \\ &\le \frac{1}{q_\ell}\|\boldsymbol{e}\|_\infty + \frac{h+1}{2} = \frac{1}{q_\ell}\mathtt{Err}_{\mathsf{sk}}(\mathsf{ct}, \boldsymbol{m}) + \frac{h+1}{2}\end{aligned}$$

as $\boldsymbol{s}$ is a ternary vector with Hamming weight h. □

Proof (Proof of Theorem 2.). In Lemmas 4 to 8, we provided specific formulas to calculate the error bounds following each homomorphic operation. These calculations rely on several parameters: the error and message bounds of the input ciphertext(s) of the operation, the error bounds of the public keys, and the Hamming weight bound of the secret key. With these parameters, and given the circuit, one can deduce the error in the output ciphertext by applying these formulas systematically through the circuit. In other words, there is an algorithm $\mathtt{Estim}(\mathtt{C}; h, B_{\mathsf{rlk}}, B_{\mathsf{rtk}}, B_{\mathsf{cjk}}, B_e, B_m)$ that calculates the upper limit of the error bound for the output ciphertext of the specified circuit $\mathtt{C}$. □

B PIOP Compilation

B.1 Interactive Argument of Knowledge

Definition 13 (Interactive Argument of Knowledge). *Let* $\Pi = (\mathtt{Setup}, \mathtt{P}, \mathtt{V})$ *be an interactive protocol between a prover* $\mathtt{P}$ *and a verifier* $\mathtt{V}$*.* Π *is called an argument of knowledge for a relation* R *if it satisfies the following properties.*

Completeness. *For all PPT adversary* $\mathtt{A}$*, the following holds.*

$$\Pr\left[\begin{array}{r|l}\langle \mathtt{P}(\mathsf{pp}, \mathsf{x}, \mathsf{w}), \mathtt{V}(\mathsf{pp}, \mathsf{x})\rangle = 1 \vee & \mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda) \\ (\mathsf{x}, \mathsf{w}) \notin \mathsf{R} & (\mathsf{x}, \mathsf{w}) \leftarrow \mathtt{A}(\mathsf{pp})\end{array}\right] \ge 1 - \mathsf{negl}(\lambda)$$

Soundness. *For every PPT adversary* $\mathtt{A} = (\mathtt{A}_1, \mathtt{A}_2)$*, the following holds.*

$$\Pr\left[\begin{array}{r|l} \langle \mathtt{A}_2(\mathsf{pp}, \mathsf{st}, \mathsf{x}), \mathtt{V}(\mathsf{pp}, \mathsf{x})\rangle = 1 \wedge & \mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda) \\ \mathsf{x} \notin \mathsf{L}(\mathsf{R}) & (\mathsf{st}, \mathsf{x}) \leftarrow \mathtt{A}_1(\mathsf{pp}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Knowledge Soundness. *For every PPT adversary* $\mathtt{A} = (\mathtt{A}_1, \mathtt{A}_2)$*, there exists a PPT extractor* $\mathtt{E}$ *such that, given oracle access to* $\mathtt{A}$*, the following holds.*

$$\Pr\left[\begin{array}{r|l} \langle \mathtt{A}_2(\mathsf{pp}, \mathsf{st}, \mathsf{x}), \mathtt{V}(\mathsf{pp}, \mathsf{x})\rangle = 1 \wedge & \mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda) \\ (\mathsf{x}, \mathsf{w}) \notin \mathsf{R} & (\mathsf{st}, \mathsf{x}) \leftarrow \mathtt{A}_1(\mathsf{pp}) \\ & \mathsf{w} \leftarrow \mathtt{E}^{\mathtt{A}}(\mathsf{pp}, \mathsf{x}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Π *is called honest verifier zero-knowledge (HVZK) if the following holds.*

Honest Verifier Zero-knowledge. *For every PPT adversary* $\mathtt{A} = (\mathtt{A}_1, \mathtt{A}_2)$*, there exists a PPT simulator* $\mathtt{S}$ *such that the following holds, where* $\mathtt{View}$ *outputs the verifier's view.*

$$\left|\Pr\left[\begin{array}{r|l} \mathtt{A}_2(\mathsf{ck}, \mathsf{view}) = 1 \wedge & \mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda) \\ (\mathsf{x}, \mathsf{w}) \in \mathsf{R} & (\mathsf{x}, \mathsf{w}) \leftarrow \mathtt{A}_1(\mathsf{pp}) \\ & \mathsf{view} \leftarrow \mathtt{S}(\mathsf{pp}, \mathsf{x}) \end{array}\right] - \Pr\left[\begin{array}{r|l} \mathtt{A}_2(\mathsf{ck}, \mathsf{view}) = 1 \wedge & \mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda) \\ (\mathsf{x}, \mathsf{w}) \in \mathsf{R} & (\mathsf{x}, \mathsf{w}) \leftarrow \mathtt{A}_1(\mathsf{pp}) \\ & \mathsf{view} \leftarrow \mathtt{View}\big(\mathtt{P}(\mathsf{pp}, \mathsf{x}, \mathsf{w}), \mathtt{V}(\mathsf{pp}, \mathsf{x})\big) \end{array}\right]\right| \leq \mathsf{negl}(\lambda)$$

Additionally, Π *is called public coin if all messages from the honest verifier can be computed as a deterministic function of a random public input.*

B.2 Polynomial Commitment Scheme

A polynomial commitment scheme (PCS) is a class of commitment schemes that takes polynomials as messages and allows the evaluation of committed polynomials. Below, we define a polynomial commitment scheme for univariate polynomials, adapted from [CHM+20].

Definition 14 (Polynomial Commitment). *A polynomial commitment* $\mathtt{PC}$ *consists of the following PPT algorithms.*

- $\underline{\mathtt{PC.Setup}}(1^\lambda, D) \to \mathsf{ck}$*: Given a security parameter* λ *and a global polynomial degree upper bound* D*, it generates a commitment key* ck*.*
- $\underline{\mathtt{PC.Com}}(\mathsf{ck}, d, \boldsymbol{f}) \to (c, \delta)$*: Given a polynomial* $\boldsymbol{f} \in \mathbb{F}[X]$ *with degree* $< d$*, it generates a commitment* c *and an opening hint* δ*.*
- $\underline{\mathtt{PC.Open}}(\mathsf{ck}, c, d, \boldsymbol{f}, \delta) \to b$*: Given a commitment* c*, a polynomial* $\boldsymbol{f}$ *with degree* $< d$*, and an opening hint* δ*, it outputs* 0 *or* 1*.*
- $\underline{\mathtt{PC.Eval}}(\mathsf{ck}, x, d, \boldsymbol{f}, \delta) \to (y, \rho)$*: Given an evaluation point* $x \in \mathbb{F}$ *and an opening hint* δ*, it returns an evaluation result* y*, and an evaluation proof* ρ*.*

– $\underline{\mathtt{PC.Check}}(\mathsf{ck}, c, d, x, y, \rho) \to b$: *Given a commitment c, a degree upper bound d, an evaluation point x, an evaluation result y, and an evaluation proof ρ, it outputs 0 or 1.*

$\mathtt{PC}$ *is called a polynomial commitment scheme if it satisfies the following properties.*

Correctness. *For every polynomial $\boldsymbol{f} \in \mathbb{F}[X]$ with a degree upper bound $d \leq D$ and every point $x \in \mathbb{F}$, the following holds.*

$$\Pr\left[\begin{array}{r|r} \mathtt{PC.Open}(\mathsf{ck}, c, d, \boldsymbol{f}, \delta) = 1 \wedge & \mathsf{ck} \leftarrow \mathtt{PC.Setup}(1^\lambda, D) \\ \mathtt{PC.Check}(\mathsf{ck}, c, d, x, \boldsymbol{f}(x), \rho) = 1 & (c, \delta) \leftarrow \mathtt{PC.Com}(\mathsf{ck}, d, \boldsymbol{f}) \\ & (y, \rho) \leftarrow \mathtt{PC.Eval}(x, \delta) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Extractability. *For every PPT adversary $\mathtt{A}$, there exists a PPT extractor $\mathtt{E}$ such that for all randomness r, the following holds.*

$$\Pr\left[\begin{array}{r|r} \mathtt{PC.Check}(\mathsf{ck}, c, d, x, y, \rho) = 1 \wedge & \mathsf{ck} \leftarrow \mathtt{PC.Setup}(1^\lambda, D) \\ \big(\mathtt{PC.Open}(\mathsf{ck}, c, d, \boldsymbol{f}, \delta) = 0 \vee & (c, d, x, y, \rho) \leftarrow \mathtt{A}(\mathsf{ck}, \mathsf{r}) \\ y \neq \boldsymbol{f}(x)\big) & (\boldsymbol{f}, \delta) \leftarrow \mathtt{E}(\mathsf{ck}, \mathsf{r}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Binding. *For every PPT adversary $\mathtt{A}$, the following holds.*

$$\Pr\left[\begin{array}{r|r} \mathtt{PC.Open}(\mathsf{ck}, c, d, \boldsymbol{f}, \delta) = 1 \wedge & \mathsf{ck} \leftarrow \mathtt{PC.Setup}(1^\lambda, D) \\ \mathtt{PC.Open}(\mathsf{ck}, c, d, \boldsymbol{f}', \delta') = 1 \wedge & (c, d, \boldsymbol{f}, \boldsymbol{f}', \delta, \delta') \leftarrow \mathtt{A}(\mathsf{ck}) \\ \boldsymbol{f} \neq \boldsymbol{f}' & \end{array}\right] \leq \mathsf{negl}(\lambda)$$

$\mathtt{PC}$ *is called hiding if the following property holds.*

Hiding. *For every PPT adversary $\mathtt{A} = (\mathtt{A}_1, \mathtt{A}_2)$, there exists a PPT simulator $\mathtt{S}$ such that the following holds.*

$$\left|\Pr\left[\begin{array}{r|r} \mathtt{A}_2(\mathsf{ck}, c, \rho) = 1 \wedge & \mathsf{ck} \leftarrow \mathtt{PC.Setup}(1^\lambda, D) \\ \mathtt{PC.Check}(\mathsf{ck}, c, d, x, \boldsymbol{f}(x), \rho) = 1 & (d, \boldsymbol{f}, x) \leftarrow \mathtt{A}_1(\mathsf{ck}) \\ & (c, \rho) \leftarrow \mathtt{S}(\mathsf{ck}, x, \boldsymbol{f}(x)) \end{array}\right]\right.$$
$$\left. - \Pr\left[\begin{array}{r|r} \mathtt{A}_2(\mathsf{ck}, c, \rho) = 1 \wedge & \mathsf{ck} \leftarrow \mathtt{PC.Setup}(1^\lambda, D) \\ \mathtt{PC.Check}(\mathsf{ck}, c, d, x, \boldsymbol{f}(x), \rho) = 1 & (d, \boldsymbol{f}, x) \leftarrow \mathtt{A}_1(\mathsf{ck}) \\ & (c, \delta) \leftarrow \mathtt{PC.Com}(\mathsf{ck}, d, \boldsymbol{f}) \\ & (y, \rho) \leftarrow \mathtt{PC.Eval}(x, \delta) \end{array}\right]\right| \leq \mathsf{negl}(\lambda)$$

In [CHM+20], Chiesa et al. formalize how a PIOP can be compiled into an argument of knowledge using a PCS. In short, the compilation is done by replacing all the oracle polynomials in the PIOP with commitments from the PCS, and then attaching evaluation proofs from the PCS for each polynomial query in the PIOP. The complexity of the resulting argument of knowledge can be described as follows.

Theorem 12 ([CHM+20, Theorem 8.1]). *Let Π be a PIOP for a relation R and $\mathtt{PC}$ be a polynomial commitment scheme. Then, there exists a public coin argument of knowledge Π' for R with the following complexity.*

Prover complexity. *The sum of the runtime of the PIOP prover, the time to commit polynomials in $\mathtt{PC}$, and the time to produce evaluation proofs for oracle queries in $\mathtt{PC}$.*

Verifier complexity. *The sum of the runtime of the PIOP verifier, the time to verify evaluation proofs in $\mathtt{PC}$.*

Proof size. *The sum of the messages from the PIOP verifier, commitments size in $\mathtt{PC}$, and evaluation proof size in $\mathtt{PC}$. Additionally, if Π is HVZK and $\mathtt{PC}$ is hiding, then Π' is HVZK.*

C Deferred PIOPs

Theorem 13. *Let $\hat{\boldsymbol{a}}_i \leftarrow \mathtt{REcd}(\vec{a}_i)$ and $\hat{\boldsymbol{b}}_i \leftarrow \mathtt{REcd}(\vec{b}_i)$ for $0 \le i < k$, where $\vec{b}_i = M_i\vec{a}_i$. Then, an interactive protocol $\Pi_{\mathtt{Lin}}$ described in Fig. 10 is an HVZK PIOP with a soundness error of $\frac{O(k+N)}{p-N}$.*

$$\Pi_{\mathtt{Lin}}(M_0, \ldots, M_{k-1}; [\![\hat{\boldsymbol{a}}_0]\!], \ldots, [\![\hat{\boldsymbol{a}}_{k-1}]\!]; [\![\hat{\boldsymbol{b}}_0]\!], \ldots, [\![\hat{\boldsymbol{b}}_{k-1}]\!])$$

Instance: matrices $M_i \in \mathbb{Z}_p^{N\times N}$ for $0 \le i < k$.
Witness: vectors $\vec{a}_i = \mathtt{Dcd}(\hat{\boldsymbol{a}}_i)$ and $\vec{b}_i = \mathtt{Dcd}(\hat{\boldsymbol{b}}_i)$ for $0 \le i < k$, where $\hat{\boldsymbol{a}}_i, \hat{\boldsymbol{b}}_i \in \mathbb{Z}_p^{<2N}[X]$.
Statement: $\vec{b}_i = M_i\vec{a}_i$ for $0 \le i < k$.

1. The verifier $\mathtt{V}$ sends random point $v, \gamma \leftarrow \mathcal{U}(\mathbb{Z}_p \setminus \mathbb{H})$.
2. The prover $\mathtt{P}$ and the verifier $\mathtt{V}$ invoke the following PIOP, where $\vec{v} = (1, v, \ldots, v^{N-1})$ and $\vec{w}_i = M_i^\top \vec{v}$.

$$\Pi_{\mathtt{IP}}\left(\sum_{i=0}^{k-1} \gamma^i \cdot (\vec{w}_i \odot \vec{X}_i - \vec{v} \odot \vec{X}_{i+k}); [\![\hat{\boldsymbol{a}}_0]\!], \ldots, [\![\hat{\boldsymbol{a}}_{k-1}]\!], [\![\hat{\boldsymbol{b}}_0]\!], \ldots, [\![\hat{\boldsymbol{b}}_{k-1}]\!]\right)$$

Fig. 10: PIOP for batched linear relations

Proof. We refer to Theorem 6.2 in [BSCR+19]. □

Theorem 14. *Let $\hat{\boldsymbol{a}}_i \leftarrow \mathtt{REcd}(\vec{a}_i)$ for $0 \le i < k$, where $\vec{\mathsf{C}}(\vec{a}_0, \ldots, \vec{a}_{k-1}) = \vec{0}$. Then, an interactive protocol $\Pi_{\mathtt{AC}}$ described in Fig. 11 is an HVZK PIOP with a soundness error of $\frac{O(dN)}{p-N}$.*

$$\Pi_{\mathtt{AC}}(\vec{\mathsf{C}}; [\![\hat{\boldsymbol{a}}_0]\!], \ldots, [\![\hat{\boldsymbol{a}}_{k-1}]\!])$$

Instance: k-ary circuit $\vec{\mathtt{C}} \in (\mathbb{Z}_p^N)[\vec{X}_0, \ldots, \vec{X}_{k-1}]$ of degree d.
Witness: vectors $\vec{a}_i = \mathtt{Dcd}(\hat{\boldsymbol{a}}_i)$ for $0 \le i < k$, where $\hat{\boldsymbol{a}}_i \in \mathbb{Z}_p^{\le 2N}[X]$.
Statement: $\vec{\mathtt{C}}(\vec{a}_0, \ldots, \vec{a}_{k-1}) = 0$.

1. Both $\mathtt{P}$ and $\mathtt{V}$ compute $\mathbf{C} \in (\mathbb{Z}_p[X])[\boldsymbol{X}_0, \ldots, \boldsymbol{X}_{k-1}]$ of degree d, which is obtained by applying $\mathtt{Ecd}$ to each coefficient of $\vec{\mathtt{C}}$.
2. The prover $\mathtt{P}$ computes the quotient polynomial $\boldsymbol{q} = \mathbf{C}(\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_{k-1})/\boldsymbol{z}_{\mathbb{H}}$, and sends a polynomial oracle $(\llbracket \boldsymbol{q} \rrbracket, d(2N-1)-1)$ to $\mathtt{V}$.
3. The verifier $\mathtt{V}$ gets evaluations $\boldsymbol{a}_i(\alpha)$ and $\boldsymbol{q}(\alpha)$ by accessing polynomial oracles $\llbracket \hat{\boldsymbol{a}}_i \rrbracket$, $\llbracket \boldsymbol{q} \rrbracket$ at a random point $\alpha \leftarrow \mathcal{U}(\mathbb{Z}_p \setminus \mathbb{H})$. Then, $\mathtt{V}$ checks whether the following holds.
$$\boldsymbol{q}(\alpha) \cdot \boldsymbol{z}_{\mathbb{H}}(\alpha) = \big(\mathbf{C}(\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_{k-1})\big)(\alpha)$$

Fig. 11: PIOP for arithmetic constraints

Proof. We refer to Theorem 4 in [HLSS25]. □

Theorem 15. *Let $\hat{\boldsymbol{a}} \leftarrow \mathtt{REcd}(\vec{a})$, where $\|\vec{a}\|_\infty \le B$. Then, an interactive protocol Π_{L^∞} described in Fig. 12 is an HVZK PIOP with a soundness error of $\frac{O(k+\ell+N)}{p-N}$.*

$$\Pi_{L^\infty}(B; \llbracket \hat{\boldsymbol{a}}_0 \rrbracket, \ldots, \llbracket \hat{\boldsymbol{a}}_{k-1} \rrbracket)$$

Instance: norm bound B.
Witness: vectors $\vec{a}_i = \mathtt{Dcd}(\hat{\boldsymbol{a}}_i)$, where $\hat{\boldsymbol{a}}_i \in \mathbb{Z}_p^{\le 2N}[X]$ for $0 \le i < k$.
Statement: $\|\vec{a}_i\|_\infty \le B$ for $0 \le i < k$.

1. The prover $\mathtt{P}$ and the verifier $\mathtt{V}$ decompose B into $B_0 = \lfloor \frac{B}{2} \rceil$, $B_1 = \lfloor \frac{B-B_0}{2} \rceil$, $B_2 = \lfloor \frac{B-B_0-B_1}{2} \rceil, \ldots, B_{\ell-1} = 1$, where $\ell = \lfloor \log B \rfloor + 1$.
2. The prover $\mathtt{P}$ decompose $\vec{a}_i$ into $\vec{a}_{i,0}, \ldots, \vec{a}_{i,\ell-1}$ such that $\vec{a}_i = \sum_{j=0}^{\ell-1} B_j \cdot \vec{a}_{i,j}$ and $\|\vec{a}_{i,j}\|_\infty \le 1$, and samples polynomial encodings $\hat{\boldsymbol{a}}_{i,j} \leftarrow \mathtt{REcd}(\vec{a}_{i,j})$. Then, $\mathtt{P}$ sends polynomial oracles $(\llbracket \hat{\boldsymbol{a}}_{i,j} \rrbracket, 2N-1)$ to $\mathtt{V}$ for $0 \le i < k$ and $0 \le j < \ell$.
3. The verifier $\mathtt{V}$ sends random points $\beta, \gamma \leftarrow \mathcal{U}(\mathbb{Z}_p \setminus \mathbb{H})$.
4. The prover $\mathtt{P}$ and the verifier $\mathtt{V}$ invoke the following PIOPs.

$$\Pi_{\mathtt{Ac}}\Bigg(\sum_{i=0}^{k-1} \gamma^i \cdot \Big(\vec{X}_{i(\ell+1)} - \sum_{j=0}^{\ell-1} B_j \cdot \vec{X}_{i(\ell+1)+j+1} + \sum_{j=0}^{\ell-1} \beta^{j+1} \cdot (\vec{X}^3_{i(\ell+1)+j+1} - \vec{X}_{i(\ell+1)+j+1}) \Big);$$
$$\llbracket \boldsymbol{a}_0 \rrbracket, \llbracket \boldsymbol{a}_{0,0} \rrbracket, \ldots, \llbracket \boldsymbol{a}_{0,\ell-1} \rrbracket, \ldots, \llbracket \boldsymbol{a}_{k-1} \rrbracket, \llbracket \boldsymbol{a}_{k-1,0} \rrbracket, \ldots, \llbracket \boldsymbol{a}_{k-1,\ell-1} \rrbracket \Bigg)$$

Fig. 12: PIOP for L^∞-norm constraints

Proof. We refer to Theorem 5 in [HLSS25]. □

D Deferred Proofs

D.1 Proof of Theorem 3

Proof. We first note that the statement $\left\langle \vec{\mathtt{C}}(\vec{a}_0, \ldots, \vec{a}_{k-1}), \vec{1} \right\rangle = 0$ is equivalent to $\sum_{h \in \mathbb{H}} \big(\mathtt{C}(\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_{k-1})\big)(h) = 0$ due to polynomial encoding.

Completeness. If $\sum_{h \in \mathbb{H}} \big(\mathtt{C}(\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_{k-1})\big)(h) = 0$, then by Lemma 1, there exist polynomials $\boldsymbol{q}$ and $\boldsymbol{r}$ such that $\boldsymbol{q} \cdot \boldsymbol{z}_{\mathbb{H}} + \boldsymbol{r} \cdot X + N^{-1} \cdot \mu = \boldsymbol{g} + \beta \cdot \mathtt{C}(\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_{k-1})$. Thus, the protocol is perfectly complete for valid witnesses.

Soundness. By Schwart-Zippel lemma, $\boldsymbol{q}(\alpha) \cdot \boldsymbol{z}_{\mathbb{H}}(\alpha) + \boldsymbol{r}(\alpha) \cdot \alpha + N^{-1} \cdot \mu = \boldsymbol{g}(\alpha) + \beta \cdot \big(\mathtt{C}(\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_{k-1})\big)(\alpha)$ holds implies $\boldsymbol{q}$ and $\boldsymbol{r}$ such that $\boldsymbol{q} \cdot \boldsymbol{z}_{\mathbb{H}} + \boldsymbol{r} \cdot X + N^{-1} \cdot \mu = \boldsymbol{g} + \beta \cdot \mathtt{C}(\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_{k-1})$, except with probability at most $O(dN)/(p - N)$. Thus, soundness holds with a soundness error of at most $\leq O(dN)/(p - N)$.

HVZK. It suffices to simulate the values of α, β, μ, $\hat{\boldsymbol{a}}_i(\alpha)$, $\{\boldsymbol{g}(\alpha)\}_{0 \leq i < k}$, $\boldsymbol{q}(\alpha)$, and $\boldsymbol{r}(\alpha)$ for $0 \leq i < k$. The simulator operates as follows.

1. Sample a random polynomial $\boldsymbol{g}'$ of degree $\leq d(2N-1)+N-1$, and decompose it into $\boldsymbol{q} \cdot \boldsymbol{z}_{\mathbb{H}} + \boldsymbol{r} \cdot X + N^{-1} \cdot \mu$.
2. Sample random polynomials $\hat{\boldsymbol{a}}_i$ of degree $< 2N$ for $0 \leq i < k$, and $\alpha, \beta \leftarrow \mathcal{U}(\mathbb{Z}_p \setminus \mathbb{H})$.
3. Set $\boldsymbol{g}(\alpha) = \boldsymbol{q}(\alpha) \cdot \boldsymbol{z}_{\mathbb{H}}(\alpha) + \boldsymbol{r}(\alpha) \cdot \alpha + N^{-1} \cdot \mu - \beta \cdot \big(\mathtt{C}(\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_{k-1})\big)(\alpha)$.
4. Output α, β, μ, $\{\hat{\boldsymbol{a}}_i(\alpha)\}_{0 \leq i < k}$, $\boldsymbol{g}(\alpha)$, $\boldsymbol{q}(\alpha)$, and $\boldsymbol{r}(\alpha)$.

The distribution of the above simulator is identical to the honest verifier's view, because the distributions of $\boldsymbol{q}, \boldsymbol{r}$, μ follow the same distribution as in the actual protocol due to random masking by $\boldsymbol{g}$, $\{\hat{\boldsymbol{a}}_i(\alpha)\}_{0 \leq i < k}$ appear uniformly random due to bounded independence, and $\boldsymbol{g}(\alpha)$ is fully determined by these values. □

D.2 Proof of Theorem 4

Proof. We first show that the statement $0 \leq [\|\vec{a}_i\|_2^2]_p \leq B$ is equivalent to $\left\langle \vec{a}_i \odot \vec{a}_i - \vec{b} \odot \vec{u}_i, \vec{1} \right\rangle = 0 \wedge \vec{u} \odot (\vec{u}_i - \vec{c}) = \vec{0}$ for some $\vec{u}_i \in \mathbb{Z}_p^N$ for $0 \leq i < k$. Suppose $0 \leq [\|\vec{a}_i\|_2^2]_p \leq B$ holds, which implies $\langle \vec{a}_i, \vec{a}_i \rangle = U_i \pmod p$ for some $0 \leq U_i \leq B$. For such U_i, there exist $u_{i,0}, \ldots, u_{i,\ell-1} \in \{0, 1\}$ such that $U_i = \sum_{j=0}^{\ell-1} u_{i,j} B_j$. Then, for $\vec{u}_i = (u_{i,0}, \ldots, u_{i,\ell-1}, 0, \ldots, 0) \in \mathbb{Z}_p^N$, it holds that $\langle \vec{a}_i, \vec{a}_i \rangle = \left\langle \vec{u}_i, \vec{b} \right\rangle \pmod p \wedge \vec{u}_i \odot (\vec{u}_i - \vec{c}) = \vec{0}$, and note that $\langle \vec{a}_i, \vec{a}_i \rangle = \left\langle \vec{u}_i, \vec{b} \right\rangle \iff \left\langle \vec{a}_i \odot \vec{a}_i, \vec{1} \right\rangle = \left\langle \vec{u}_i \odot \vec{b}, \vec{1} \right\rangle \iff \left\langle \vec{a}_i \odot \vec{a}_i - \vec{u}_i \odot \vec{b}, \vec{1} \right\rangle$.

Conversely, suppose there exists $\vec{u}_i \in \mathbb{Z}_p^N$ such that $\left\langle \vec{a}_i \odot \vec{a}_i - \vec{b} \odot \vec{u}_i, \vec{1} \right\rangle = 0 \wedge \vec{u}_i \odot (\vec{u}_i - \vec{c}) = \vec{0}$. Then, $\vec{u}_i$ is of the form $(u_{i,0}, \ldots, u_{i,\ell-1}, 0, \ldots, 0) \in \mathbb{Z}_p^N$, and $\langle \vec{a}_i, \vec{a}_i \rangle = \sum_{j=0}^{\ell-1} u_{i,j} B_j \pmod p$. Note that $0 \leq \sum_{j=0}^{\ell-1} u_{i,j} B_j \leq B$ holds, so we have $0 \leq [\|\vec{a}_i\|_2^2]_p \leq B$.

Finally, by the Schwartz-Zippel lemma, $\sum_{i=0}^{k-1} \gamma^i \cdot \left\langle \vec{a}_i \odot \vec{a}_i - \vec{b} \odot \vec{u}_i, \vec{1} \right\rangle = 0 \wedge \sum_{i=0}^{k-1} \gamma^i \cdot \vec{u}_i \odot (\vec{u}_i - \vec{c}) = \vec{0}$ implies $\left\langle \vec{a}_i \odot \vec{a}_i - \vec{b} \odot \vec{u}_i, \vec{1} \right\rangle = 0 \wedge \vec{u}_i \odot (\vec{u}_i - \vec{c}) = \vec{0}$ for $0 \leq i < k$ except probability at most $k/(p-N)$. We note that the completeness, knowledge soundness, and HVZK properties directly follow from Theorems 3 and 14. □

D.3 Proof of Theorem 5

Proof. Let $\mathtt{A}$ be a PPT adversary that manipulate the receiver and $\mathtt{S}$ be a PPT simulator for $\mathtt{A}$. Then, for all PPT environments $\mathtt{Z}$, $\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}, \mathtt{Z}]$ is an ε_2-DP prediction with respect to θ, since in the distribution of $\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}, \mathtt{Z}]$, only the output $\boldsymbol{y} \leftarrow \mathtt{M}(\theta, \vec{\boldsymbol{x}})$ obtained from $\mathtt{F}_\theta$ varies with the choice of θ, and $\mathtt{M}(\theta, \vec{\boldsymbol{x}})$ is an ε_2-DP prediction.

Now, let $\theta, \theta' \in \Theta$ be adjacent, then we obtain the followings for all PPT algorithm $\mathtt{D}$.

$$\begin{aligned}
\Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_\theta, \mathtt{A}, \mathtt{Z}]) = 1\right] &\leq e^{\varepsilon_1} \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}, \mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda) \\
&\leq e^{\varepsilon_1 + \varepsilon_2} \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_{\theta'}, \mathtt{S}, \mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda) \\
&\leq e^{2\varepsilon_1 + \varepsilon_2} \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_{\theta'}, \mathtt{A}, \mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda)
\end{aligned}$$

Therefore, $\mathtt{EXEC}[\Pi_\theta, \mathtt{A}, \mathtt{Z}]$ is a $(2\varepsilon_1 + \varepsilon_2)$-DP prediction with respect to θ. □

D.4 Proof of Theorem 6

Proof. Let $\mathtt{A}$ be a PPT adversary that manipulates the receiver in Π and $\mathtt{Z}$ be a PPT environment. Then, it holds that $\mathtt{EXEC}[\Pi_\theta, \mathtt{A}, \mathtt{Z}] \approx_c \mathtt{EXEC}[\Pi_\theta, \mathtt{A}^*, \mathtt{A}|\mathtt{Z}]$, where $\mathtt{A}|\mathtt{Z}$ is an environment that both runs $\mathtt{A}$ and $\mathtt{Z}$. By assumption, there exists a simulator $\mathtt{S}^*$ for the dummy adversary where the following holds for all PPT algorithms $\mathtt{D}$.

$$\Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_\theta, \mathtt{A}^*, \mathtt{A}|\mathtt{Z}]) = 1\right] \leq e^{\varepsilon} \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}^*, \mathtt{A}|\mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda)$$

$$\Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}^*, \mathtt{A}|\mathtt{Z}]) = 1\right] \leq e^{\varepsilon} \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_\theta, \mathtt{A}^*, \mathtt{A}|\mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda)$$

Additionally, it holds that $\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}^*, \mathtt{A}|\mathtt{Z}] \approx_c \mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}^*|\mathtt{A}, \mathtt{Z}]$, where $\mathtt{S}^*|\mathtt{A}$ is a PPT algorithm that runs both $\mathtt{S}^*$ and $\mathtt{A}$. Then, we have the followings for all PPT algorithms $\mathtt{D}$.

$$\begin{aligned}
\Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_\theta, \mathtt{A}, \mathtt{Z}]) = 1\right] &\leq \Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_\theta, \mathtt{A}^*, \mathtt{A}|\mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda) \\
&\leq e^{\varepsilon} \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}^*, \mathtt{A}|\mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda) \\
&\leq e^{\varepsilon} \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}^*|\mathtt{A}, \mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda)
\end{aligned}$$

$$\begin{aligned}
\Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}^*|\mathtt{A}, \mathtt{Z}]) = 1\right] &\leq \Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}^*, \mathtt{A}|\mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda) \\
&\leq e^{\varepsilon} \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_\theta, \mathtt{A}^*, \mathtt{A}|\mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda) \\
&\leq e^{\varepsilon} \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_\theta, \mathtt{A}, \mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda)
\end{aligned}$$

Therefore, Π satisfies sender's privacy with the parameter ε. □

D.5 Proof of Theorem 7

Proof. Let $\mathtt{A}$ be a PPT adversary that manipulates the receiver in Π_θ for $\theta \in \Theta$. Since Π' securely implements $\mathtt{F}'$, there exists a PPT adversary $\mathtt{A}'$ such that $\mathtt{EXEC}[\Pi_\theta, \mathtt{A}, \mathtt{Z}] \approx_c \mathtt{EXEC}[\Pi_{\mathtt{F}',\theta}, \mathtt{A}', \mathtt{Z}]$ for all PPT environments $\mathtt{Z}$ by the universal composition theorem. Also, there exists a simulator $\mathtt{S}$ such that the following holds for all PPT algorithms $\mathtt{D}$ since $\Pi_{\mathtt{F}'}$ satisfies the sender's privacy with a parameter ε.

$$\Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_{\mathtt{F}',\theta}, \mathtt{A}', \mathtt{Z}]) = 1\right] \leq e^\varepsilon \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}, \mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda)$$

$$\Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}, \mathtt{Z}]) = 1\right] \leq e^\varepsilon \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_{\mathtt{F}',\theta}, \mathtt{A}', \mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda)$$

Therefore, we obtain the following since $\mathtt{EXEC}[\Pi_\theta, \mathtt{A}, \mathtt{Z}] \approx_c \mathtt{EXEC}[\Pi_{\mathtt{F}',\theta}, \mathtt{A}', \mathtt{Z}]$, which shows that Π also satisfies the sender's privacy with the parameter ε.

$$\Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_\theta, \mathtt{A}, \mathtt{Z}]) = 1\right] \leq e^\varepsilon \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}, \mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda)$$

$$\Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}, \mathtt{Z}]) = 1\right] \leq e^\varepsilon \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_\theta, \mathtt{A}, \mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda)$$

□

D.6 Proof of Theorem 8

Proof. Below, we demonstrate the usefulness, receiver's privacy, and sender's privacy of $\Pi_{\mathtt{CKKS}}$.

Usefulness. Let $\vec{\boldsymbol{x}} \in \mathcal{X}^k$ be a receiver's input, $\theta \in \Theta$ be a sender's input, and $\boldsymbol{y} \in R$ be a receiver's output. Then, the following holds for some $\boldsymbol{e}_{\mathtt{Eval}}$ such that $\|\boldsymbol{e}_{\mathtt{Eval}}\|_\infty \leq \tau$, and $\eta > \tau + 1/2$ since $\boldsymbol{t} \leftarrow \mathtt{Lap}(N\tau/\varepsilon)^N$.

$$\begin{aligned}\Pr[\|\boldsymbol{y} - \mathtt{f}(\theta, \vec{\boldsymbol{x}})\|_\infty > \eta] &= \Pr[\|\boldsymbol{e}_{\mathtt{Eval}} + \lfloor \boldsymbol{t} \rceil\|_\infty > \eta] \\ &\leq \Pr[\|\lfloor \boldsymbol{t} \rceil\|_\infty > \eta - \tau] \leq \Pr[\|\boldsymbol{t}\|_\infty > \eta - \tau - 1/2] \\ &\leq \frac{1}{2} \cdot \exp\left(-\frac{\varepsilon}{N\tau}(\eta - \tau - 1/2)\right)\end{aligned}$$

Receiver's privacy. Let $\Pi_{\mathtt{ZKAoK}}$ be a protocol that securely implements the functionality $\mathtt{F}_{\mathtt{ZKAoK}}$. Then, there exists a simulator $\mathtt{S}_{\mathtt{ZKAoK}}$ such that $\mathtt{S}_{\mathtt{ZKAoK}}(\mathsf{x}_{\mathtt{CKKS}}) \approx_c \mathtt{VIEW}_{\mathtt{P}_2}^{\Pi_{\mathtt{ZKAoK}}}(\mathsf{x}_{\mathtt{CKKS}}, \mathsf{w}_{\mathtt{CKKS}})$ for all $\vec{\boldsymbol{x}} \in \mathcal{X}$. Now, suppose there exists a PPT algorithm $\mathtt{D}$ that distinguishes $\mathtt{VIEW}_{\mathtt{P}_2}^{\Pi_{\mathtt{CKKS}}}(\vec{\boldsymbol{x}}, \theta)$ and $\mathtt{VIEW}_{\mathtt{P}_2}^{\Pi_{\mathtt{CKKS}}}(\vec{\boldsymbol{x}}', \theta)$ with non-negligible probability for some $\vec{\boldsymbol{x}}, \vec{\boldsymbol{x}}' \in \mathcal{X}$. Then, the following holds for $\mathtt{VIEW}_{\mathtt{P}_2}^{\Pi_{\mathtt{CKKS}}}(\vec{\boldsymbol{x}}, \theta)$ and $\mathtt{VIEW}_{\mathtt{P}_2}^{\Pi_{\mathtt{CKKS}}}(\vec{\boldsymbol{x}}', \theta)$.

$$\mathtt{VIEW}_{\mathtt{P}_2}^{\Pi_{\mathtt{CKKS}}}(\vec{\boldsymbol{x}}, \theta) = \left(\mathsf{x}_{\mathtt{CKKS}}, \mathtt{VIEW}_{\mathtt{P}_2}^{\Pi_{\mathtt{ZKAoK}}}(\mathsf{x}_{\mathtt{CKKS}}, \mathsf{w}_{\mathtt{CKKS}})\right) \approx_c \left(\mathsf{x}_{\mathtt{CKKS}}, \mathtt{S}_{\mathtt{ZKAoK}}(\mathsf{x}_{\mathtt{CKKS}})\right)$$

$$\mathtt{VIEW}_{\mathtt{P}_2}^{\Pi_{\mathtt{CKKS}}}(\vec{\boldsymbol{x}}', \theta) = \left(\mathsf{x}'_{\mathtt{CKKS}}, \mathtt{VIEW}_{\mathtt{P}_2}^{\Pi_{\mathtt{ZKAoK}}}(\mathsf{x}'_{\mathtt{CKKS}}, \mathsf{w}'_{\mathtt{CKKS}})\right) \approx_c \left(\mathsf{x}'_{\mathtt{CKKS}}, \mathtt{S}_{\mathtt{ZKAoK}}(\mathsf{x}'_{\mathtt{CKKS}})\right)$$

Then, $\mathtt{D}$ also distinguishes $\left(\mathsf{x}_{\mathtt{CKKS}}, \mathtt{S}_{\mathtt{ZKAoK}}(\mathsf{x}_{\mathtt{CKKS}})\right)$ and $\left(\mathsf{x}'_{\mathtt{CKKS}}, \mathtt{S}_{\mathtt{ZKAoK}}(\mathsf{x}'_{\mathtt{CKKS}})\right)$ with non-negligible probability, which contradicts the IND-CPA security of the CKKS scheme. Therefore, $\Pi_{\mathtt{CKKS}}$ achieves the receiver's privacy in the $\mathtt{F}_{\mathtt{ZKAoK}}$-hybrid model since no such algorithm $\mathtt{D}$ exists.

Sender's privacy. By Theorem 6, it suffices to consider only the dummy adversary $\mathtt{A}^*$ for all PPT environments. For a PPT environment $\mathtt{Z}$, We build a simulator $\mathtt{S}$ for the dummy adversary $\mathtt{A}^*$ as follows.

1. When $\mathtt{Z}$ sends a message $(\mathsf{pk}, \{\mathsf{ct}_0, \ldots, \mathsf{ct}_{k-1}\})$ that is supposed to be forwarded by the dummy adversary to the honest sender, the simulator $\mathtt{S}$ retrieves it.
2. When $\mathtt{Z}$ sends a message $(\mathsf{x}_{\mathtt{CKKS}}, \mathsf{w}_{\mathtt{CKKS}})$ that is supposed to be forwarded to the ideal functionality $\mathtt{F}_{\mathtt{ZKAoK}}$, the simulator $\mathtt{S}$ retrieves it.
3. If $(\mathsf{x}_{\mathtt{CKKS}}, \mathsf{w}_{\mathtt{CKKS}}) \in \mathsf{R}_{\mathtt{CKKS}}$, the simulator $\mathtt{S}$ continues the protocol, otherwise it aborts.
4. The simulator $\mathtt{S}$ sends $\vec{\boldsymbol{x}}$ to the functionality $\mathtt{F}_\theta$, and receives an output $\boldsymbol{y}'$ from the functionality.
5. The simulator $\mathtt{S}$ samples $\boldsymbol{a}' \leftarrow \mathcal{U}(R_{Q_1})$, and $\boldsymbol{r}_0 \leftarrow \mathtt{DG}(\sigma)^N, \boldsymbol{r}_1 \leftarrow \mathtt{DG}(B_{\mathsf{ek}} N \cdot \sigma)^N, \boldsymbol{r}_2 \leftarrow \mathtt{DG}(h\sigma)^N, \boldsymbol{r}_3 \leftarrow \mathtt{DG}(\sigma)^N$. Then, it returns $\mathsf{ct}'_{\mathsf{out}}$ to the environment $\mathtt{Z}$, which is computed as follows.

$$\mathsf{ct}'_{\mathsf{out}} = \Big(-\boldsymbol{a}' \cdot \boldsymbol{s} + (\boldsymbol{e}_{\mathsf{ek}} \cdot \boldsymbol{r}'_0 + \boldsymbol{r}'_1) + (\boldsymbol{r}'_2 - \boldsymbol{s} \cdot \boldsymbol{r}'_3) + q_1 \cdot \boldsymbol{y}', \boldsymbol{a}' \Big) \in R^2_{Q_1}$$

Now, we show that for all PPT algorithms $\mathtt{D}$, the following holds, which implies the sender's privacy with the parameter ε.

$$\Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_{\mathtt{CKKS}}, \mathtt{A}^*, \mathtt{Z}]) = 1\right] \leq e^\varepsilon \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}, \mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda)$$

$$\Pr\left[\mathtt{D}(\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}, \mathtt{Z}]) = 1\right] \leq e^\varepsilon \cdot \Pr\left[\mathtt{D}(\mathtt{EXEC}[\Pi_{\mathtt{CKKS}}, \mathtt{A}^*, \mathtt{Z}]) = 1\right] + \mathsf{negl}(\lambda)$$

We note that the only difference between $\mathtt{EXEC}[\Pi_{\mathtt{CKKS}}, \mathtt{A}^*, \mathtt{Z}]$ and $\mathtt{EXEC}[\mathtt{F}_\theta, \mathtt{S}, \mathtt{Z}]$ is the distribution of $\mathsf{ct}_{\mathsf{out}}$ and $\mathsf{ct}'_{\mathsf{out}}$. Thus, it suffices to prove that for all PPT algorithms $\mathtt{D}$, the following holds.

$$\Pr\left[\mathtt{D}(\mathsf{ct}_{\mathsf{out}}, \mathsf{w}_{\mathtt{CKKS}}) = 1\right] \leq e^\varepsilon \cdot \Pr\left[\mathtt{D}(\mathsf{ct}'_{\mathsf{out}}, \mathsf{w}_{\mathtt{CKKS}}) = 1\right] + \mathsf{negl}(\lambda) \tag{10}$$

$$\Pr\left[\mathtt{D}(\mathsf{ct}'_{\mathsf{out}}, \mathsf{w}_{\mathtt{CKKS}}) = 1\right] \leq e^\varepsilon \cdot \Pr\left[\mathtt{D}(\mathsf{ct}_{\mathsf{out}}, \mathsf{w}_{\mathtt{CKKS}}) = 1\right] + \mathsf{negl}(\lambda) \tag{11}$$

For the environment $\mathtt{Z}$, $\mathsf{ct}_{\mathsf{out}}$ has the following form, where $\boldsymbol{a} = q_1 \cdot \boldsymbol{c}_1 + (\mathsf{ek}_0 \cdot \boldsymbol{r}_0 + \boldsymbol{r}_3)$.

$$\mathsf{ct}_{\mathsf{out}} = \Big(-\boldsymbol{a} \cdot \boldsymbol{s} + (\boldsymbol{e}_{\mathsf{ek}} \cdot \boldsymbol{r}_0 + \boldsymbol{r}_1) + (\boldsymbol{r}_2 - \boldsymbol{s} \cdot \boldsymbol{r}_3) + q_1 \cdot \boldsymbol{y}, \boldsymbol{a} \Big) \in R^2_{Q_1}$$

Let $\mathsf{ct}''_{\mathsf{out}}$ be a hybrid distribution defined as follows, where $\boldsymbol{a}' \leftarrow \mathcal{U}(R_{Q_1})$.

$$\mathsf{ct}''_{\mathsf{out}} = \Big(-\boldsymbol{a}' \cdot \boldsymbol{s} + (\boldsymbol{e}_{\mathsf{ek}} \cdot \boldsymbol{r}_0 + \boldsymbol{r}_1) + (\boldsymbol{r}_2 - \boldsymbol{s} \cdot \boldsymbol{r}_3) + q_1 \cdot \boldsymbol{y}, \boldsymbol{a}' \Big) \in R^2_{Q_1}$$

Then, $\mathsf{ct}_{\mathsf{out}} \approx_c \mathsf{ct}''_{\mathsf{out}}$ assuming that $\mathsf{HintRLWE}^{\boldsymbol{e}_{\mathsf{ek}}, \boldsymbol{s}, B_{\mathsf{ek}} N\sigma, h\sigma}_{Q_1, \sigma, \sigma}$ is computationally hard, which can be reduced from $\mathsf{RLWE}_{Q_1, \sigma/2, \sigma/2}$ with an advantage of at most $\mathsf{negl}(\lambda)$ by Theorem 1.

For $\mathsf{ct}''_{\mathsf{out}}$ and $\mathsf{ct}'_{\mathsf{out}}$, they only differ in the distribution of $\boldsymbol{y}$ and $\boldsymbol{y}'$. We note that $\boldsymbol{y} = \mathtt{f}(\theta, \vec{\boldsymbol{x}}) + \boldsymbol{e}_{\mathtt{Eval}} + \lfloor \boldsymbol{t} \rceil$ and $\boldsymbol{y}' = \mathtt{f}(\theta, \vec{\boldsymbol{x}}) + \lfloor \boldsymbol{t}' \rceil$ hold for some $\boldsymbol{e}_{\mathtt{Eval}}$, where $\boldsymbol{t}, \boldsymbol{t}' \leftarrow$

$\mathtt{Lap}(N\tau/\varepsilon)^N$ and $\|\boldsymbol{e}_{\mathtt{Eval}}\|_\infty \le \tau$, which implies $\|\boldsymbol{e}_{\mathtt{Eval}}\|_1 \le N\tau$. Thus, $D_\infty(\boldsymbol{y}\|\boldsymbol{y}') \le \varepsilon$ holds due to the Laplacian mechanism, which implies $D_\infty(\mathsf{ct}''_{\mathsf{out}}\|\mathsf{ct}'_{\mathsf{out}}) \le \varepsilon$, where D_∞ denotes the max-divergence. Therefore, Eqs. (10) and (11) holds since $\mathsf{ct}_{\mathsf{out}} \approx_c \mathsf{ct}''_{\mathsf{out}}$. □

D.7 Proof of Theorem 11

Proof. Since $R_p \cong \mathbb{Z}_p^N$ via an isomorphism $\mathtt{NTT}$, we can prove the well-formedness of public keys and ciphertexts in the NTT representation, and norm constraints for the witness in the coefficient representation. Thus, the conditions for $\mathsf{R}'_{\mathtt{CKKS}}$ are directly translated from Eq. (2) to Eq. (9), except for the conditions for the secret key $\boldsymbol{s}$, where we need to prove $\|\boldsymbol{s}\|_\infty \le 1 \wedge \|\boldsymbol{s}\|_1 \le h$.

To be precise, in Eq. (1), we prove that $\|\boldsymbol{s}\|_\infty \le 1 \wedge [\|\boldsymbol{s}\|_2^2]_p \le h$. However, this statement is equivalent to $\|\boldsymbol{s}\|_\infty \le 1 \wedge \|\boldsymbol{s}\|_1 \le h$ since $[\|\boldsymbol{s}\|_2^2]_p = \|\boldsymbol{s}\|_1$ under the condition $\|\boldsymbol{s}\|_\infty \le 1$. Therefore, $\Pi_{\mathtt{CT\&PK}}$ proves the conditions in $\mathsf{R}'_{\mathtt{CKKS}}$. We note that the completeness, knowledge soundness, and zero-knowledge properties directly follow from Theorems 4 and 13 to 15. □

D.8 Proof of Theorem 10

Proof. We show how the norm bound changes for each type of public key and ciphertext after modulus switching.

For the encryption key, it holds that $\mathsf{ek} = \left(\left\lfloor \frac{Q_L}{p} \cdot \mathsf{ek}'_0 \right\rceil, \left\lfloor \frac{Q_L}{p} \cdot \mathsf{ek}'_1 \right\rceil\right)$. Then, we have the following error terms $\boldsymbol{\varepsilon}_{\mathsf{ek},0}, \boldsymbol{\varepsilon}_{\mathsf{ek},1} \in \mathbb{R}[X]/(X^N+1)$.

$$\begin{aligned}\left\lfloor \frac{Q_L}{p} \cdot \mathsf{ek}'_0 \right\rceil + \left\lfloor \frac{Q_L}{p} \cdot \mathsf{ek}'_1 \right\rceil \cdot \boldsymbol{s} &= \left(\frac{Q_L}{p} \cdot \mathsf{ek}'_0 + \boldsymbol{\varepsilon}_{\mathsf{ek},0}\right) + \left(\frac{Q_L}{p} \cdot \mathsf{ek}'_1 + \boldsymbol{\varepsilon}_{\mathsf{ek},1}\right) \cdot \boldsymbol{s} \\ &= \frac{Q_L}{p} \cdot \boldsymbol{e}'_{\mathsf{ek}} + (\boldsymbol{\varepsilon}_{\mathsf{ek},0} + \boldsymbol{\varepsilon}_{\mathsf{ek},1} \cdot \boldsymbol{s})\end{aligned}$$

Then, for $\boldsymbol{e}_{\mathsf{ek}} = \frac{Q_L}{p} \cdot \boldsymbol{e}'_{\mathsf{ek}} + \boldsymbol{\varepsilon}_{\mathsf{ek},0} + \boldsymbol{\varepsilon}_{\mathsf{ek},1} \cdot \boldsymbol{s}$, it holds that $\|\boldsymbol{e}_{\mathsf{ek}}\|_\infty \le B' + \frac{h+1}{2}$ since $Q_L < p$, $\|\boldsymbol{\varepsilon}_{\mathsf{ek},0}\|_\infty, \|\boldsymbol{\varepsilon}_{\mathsf{ek},1}\|_\infty \le 1/2$, and $\|\boldsymbol{s}\|_1 \le h$.

For the relinearization key, it holds that $\mathsf{rlk} = \left(\left\lfloor \frac{Q_L}{p} \cdot \mathsf{rlk}'_0 \right\rceil, \left\lfloor \frac{Q_L}{p} \cdot \mathsf{rlk}'_1 \right\rceil\right)$. Then, we have the following error terms $\vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},0}, \vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},1}, \vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},2} \in \mathbb{R}[X]/(X^N+1)$.

$$\begin{aligned}\left\lfloor \frac{Q_L}{p} \cdot \mathsf{rlk}'_0 \right\rceil + \left\lfloor \frac{Q_L}{p} \cdot \mathsf{rlk}'_1 \right\rceil \cdot \boldsymbol{s} &= \left(\frac{Q_L}{p} \cdot \mathsf{rlk}'_0 + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},0}\right) + \left(\frac{Q_L}{p} \cdot \mathsf{rlk}'_1 + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},1}\right) \cdot \boldsymbol{s} \\ &= \frac{Q_L}{p} \cdot (\boldsymbol{s}^2 \cdot \lfloor p/Q_L \cdot \vec{g} \rceil + \vec{\boldsymbol{e}}'_{\mathsf{rlk}}) + (\vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},0} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},1} \cdot \boldsymbol{s}) \\ &= \frac{Q_L}{p} \cdot \left(\boldsymbol{s}^2 \cdot (p/Q_L \cdot \vec{g} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},2}) + \vec{\boldsymbol{e}}'_{\mathsf{rlk}}\right) + (\vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},0} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},1} \cdot \boldsymbol{s}) \\ &= \boldsymbol{s}^2 \cdot \vec{g} + \frac{Q_L}{p} \cdot \vec{\boldsymbol{e}}'_{\mathsf{rlk}} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},0} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},1} \cdot \boldsymbol{s} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},1} \cdot \boldsymbol{s}^2 \pmod{Q_L}\end{aligned}$$

Then, for $\vec{\boldsymbol{e}}_{\mathsf{rlk}} = \frac{Q_L}{p} \cdot \vec{\boldsymbol{e}}'_{\mathsf{rlk}} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},0} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},1} \cdot \boldsymbol{s} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},2} \cdot \boldsymbol{s}^2$, it holds that $\|\vec{\boldsymbol{e}}_{\mathsf{rlk}}\|_\infty \le B' + \frac{h^2+h+1}{2}$ since $Q_L < p$, $\|\vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},0}\|_\infty, \|\vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},1}\|_\infty, \|\vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},2}\|_\infty \le 1/2$ and $\|\boldsymbol{s}\|_1 \le h$.

For the rotation key, it holds that $\mathsf{rtk} = \left(\left\lfloor \frac{Q_L}{p} \cdot \mathsf{rtk}'_0 \right\rceil, \left\lfloor \frac{Q_L}{p} \cdot \mathsf{rtk}'_1 \right\rceil\right)$. Then, we have the following error terms $\vec{\boldsymbol{\varepsilon}}_{\mathsf{rtk},0}, \vec{\boldsymbol{\varepsilon}}_{\mathsf{rtk},1}, \vec{\boldsymbol{\varepsilon}}_{\mathsf{rtk},2} \in \mathbb{R}[X]/(X^N+1)$.

$$\begin{aligned}
\left\lfloor \frac{Q_L}{p} \cdot \mathsf{rtk}'_0 \right\rceil + \left\lfloor \frac{Q_L}{p} \cdot \mathsf{rtk}'_1 \right\rceil \cdot \boldsymbol{s} &= \left(\frac{Q_L}{p} \cdot \mathsf{rtk}'_0 + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rtk},0}\right) + \left(\frac{Q_L}{p} \cdot \mathsf{rtk}'_1 + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rtk},1}\right) \cdot \boldsymbol{s} \\
&= \frac{Q_L}{p} \cdot (\varphi(\boldsymbol{s}) \cdot \lfloor p/Q_L \cdot \vec{g} \rceil + \vec{\boldsymbol{e}}'_{\mathsf{rtk}}) + (\vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},0} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rtk},1} \cdot \boldsymbol{s}) \\
&= \frac{Q_L}{p} \cdot \left(\varphi(\boldsymbol{s}) \cdot (p/Q_L \cdot \vec{g} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rtk},2}) + \vec{\boldsymbol{e}}'_{\mathsf{rtk}}\right) + (\vec{\boldsymbol{\varepsilon}}_{\mathsf{rtk},0} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rtk},1} \cdot \boldsymbol{s}) \\
&= \boldsymbol{s}^2 \cdot \vec{g} + \frac{Q_L}{p} \cdot \vec{\boldsymbol{e}}'_{\mathsf{rtk}} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rtk},0} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rtk},1} \cdot \boldsymbol{s} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rtk},2} \cdot \varphi(\boldsymbol{s}) \pmod{Q_L}
\end{aligned}$$

Then, for $\vec{\boldsymbol{e}}_{\mathsf{rtk}} = \frac{Q_L}{p} \cdot \vec{\boldsymbol{e}}'_{\mathsf{rlk}} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rtk},0} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rtk},1} \cdot \boldsymbol{s} + \vec{\boldsymbol{\varepsilon}}_{\mathsf{rtk},1} \cdot \varphi(\boldsymbol{s})$, it holds that $\|\vec{\boldsymbol{e}}_{\mathsf{rtk}}\|_\infty \leq B' + \frac{2h+1}{2}$ since $Q_L < p$, $\|\vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},0}\|_\infty, \|\vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},1}\|_\infty, \|\vec{\boldsymbol{\varepsilon}}_{\mathsf{rlk},2}\|_\infty \leq 1/2$, and $\|\varphi(\boldsymbol{s})\|_1 = \|\boldsymbol{s}\|_1 \leq h$. We note that for the conjugation key, the overall process is identical to the rotation key case since $\|\psi(\boldsymbol{s})\|_1 = \|\boldsymbol{s}\|_1$ holds, so we omit the details.

For the ciphertexts, we first analyze $\left\lfloor \frac{Q_L}{p} \cdot \mathsf{ct}'_i \right\rceil = \left(\left\lfloor \frac{Q_L}{p} \cdot \mathsf{ct}'_{i,0} \right\rceil, \left\lfloor \frac{Q_L}{p} \cdot \mathsf{ct}'_{i,1} \right\rceil\right)$. Then, we have the following for error terms $\boldsymbol{\varepsilon}_{i,0}, \boldsymbol{\varepsilon}_{i,1}, \boldsymbol{\varepsilon}_{i,2} \in \mathbb{R}[X]/(X^N+1)$, where we represent $\boldsymbol{m}'_i = \frac{p}{Q_L}\boldsymbol{m}_i + \boldsymbol{\varepsilon}_{i,2}$ for $\|\boldsymbol{m}_i\|_\infty \leq \Delta$ and $\boldsymbol{\varepsilon}_{i,2} \leq 1/2$.

$$\begin{aligned}
\left\lfloor \frac{Q_L}{p} \cdot \mathsf{ct}'_{i,0} \right\rceil + \left\lfloor \frac{Q_L}{p} \cdot \mathsf{ct}'_{i,1} \right\rceil \cdot \boldsymbol{s} &= \left(\frac{Q_L}{p} \cdot \mathsf{ct}'_{i,0} + \boldsymbol{\varepsilon}_{i,0}\right) + \left(\frac{Q_L}{p} \cdot \mathsf{ct}'_{i,1} + \boldsymbol{\varepsilon}_{i,1}\right) \cdot \boldsymbol{s} \\
&= \frac{Q_L}{p} \cdot (\boldsymbol{m}'_i + \boldsymbol{e}'_i) + (\boldsymbol{\varepsilon}_{i,0} + \boldsymbol{\varepsilon}_{i,1} \cdot \boldsymbol{s}) \\
&= \boldsymbol{m}_i + \frac{Q_L}{p} \cdot \boldsymbol{\varepsilon}_{i,2} + \frac{Q_L}{p} \cdot \boldsymbol{e}'_i + (\boldsymbol{\varepsilon}_{i,0} + \boldsymbol{\varepsilon}_{i,1} \cdot \boldsymbol{s}) \pmod{Q_L}
\end{aligned}$$

Therefore, $\left\lfloor \frac{Q_L}{p} \cdot \mathsf{ct}'_i \right\rceil \in R^2_{Q_L}$ is a ciphertext, which encrypts $\boldsymbol{m}_i$ with an error bound $\leq B' + \frac{h+3}{2}$. Now, consider $(\mathsf{ct}_{2i}, \mathsf{ct}_{2i+1}) \leftarrow \mathtt{CtoS}_{\mathsf{pk}}\left(\left\lfloor \frac{Q_L}{p} \cdot \mathsf{ct}'_i \right\rceil\right)$ for $0 \leq i < k/2$. Then, it holds the following.

$$\begin{aligned}
\mathsf{ct}_{2i,0} + \mathsf{ct}_{2i,1} \cdot \boldsymbol{s} &= \widetilde{\boldsymbol{m}}_{2i} + \widetilde{\boldsymbol{e}}_{2i} \pmod{Q_L} \\
\mathsf{ct}_{2i+1,0} + \mathsf{ct}_{2i+1,1} \cdot \boldsymbol{s} &= \widetilde{\boldsymbol{m}}_{2i+1} + \widetilde{\boldsymbol{e}}_{2i+1} \pmod{Q_L}
\end{aligned}$$

where $\iota(\widetilde{\boldsymbol{m}}_{2i}) = (m_{i,0}, \ldots, m_{i,N/2-1})$ and $\iota(\widetilde{\boldsymbol{m}}_{2i+1}) = (m_{i,N/2}, \ldots, m_{i,N-1})$ for $(m_{i,0}, \ldots, m_{i,N-1}) = \mathtt{Coeff}(\boldsymbol{m}_i)$, and $\|\widetilde{\boldsymbol{e}}_{2i}\|_\infty, \|\widetilde{\boldsymbol{e}}_{2i+1}\|_\infty \leq B_e$. Therefore, we have $\widetilde{\boldsymbol{m}}_{2i}, \widetilde{\boldsymbol{m}}_{2i+1} \in \mathcal{X}$ since $\|\mathtt{Unpack}(\widetilde{\boldsymbol{m}}_{2i})\|_\infty, \|\mathtt{Unpack}(\widetilde{\boldsymbol{m}}_{2i+1})\|_\infty \leq \Delta^{-1} \cdot \|\boldsymbol{m}\|_\infty = 1$, and $\mathtt{Unpack}(\cdot) = \Delta^{-1} \cdot \iota(\cdot)$.