

Reaction Attack on CKKS Deployments with Asymptotically Optimal Query Complexity*

Song Bian
Beihang University
sbian@buaa.edu.cn

Hyeongmin Choe
University of Luxembourg
hyeongmin.choe@uni.lu

Jiseung Kim
Jeonbuk National University
jiseungkim@jbnu.ac.kr

Changmin Lee[†]
Korea University
changminlee@korea.ac.kr

Abstract—We study a protocol-level reaction leakage in CKKS deployments where public coefficient-to-slot (C2S) evaluation keys are combined with a client’s 1-bit validity response. Under this deployment surface, a malicious evaluator can recover a ternary secret key using asymptotically optimal $O(\mathcal{H})$ reaction queries, where $\mathcal{H}$ is the entropy of the secret key.

The attack does not break RLWE or CKKS IND-CPA security. Instead, it exploits the interaction between client-side validity reactions and public homomorphic linear transformations. A shift of a public encryption key yields a ciphertext whose plaintext is a function of the secret key; the public C2S layer then repacks this information to construct a ciphertext whose validity depends strictly on whether a targeted secret coefficient matches a guessed value. Furthermore, each submitted query is independently randomized with a fresh encryption of zero, so that each resulting malicious ciphertext appears entirely distinct, thereby allowing the attack to simply evade trivial detection mechanisms on the client side.

We implemented the full end-to-end attack in Lattigo v6.2.0 and recovered the full secret key across four parameter sets drawn from standard CKKS presets. For uniform ternary secrets, the attack uses $O(N)$ queries and is within a factor $2/\log_2 3 < 2$ of the oracle-only entropy lower bound. For sparse ternary secrets, a tailored variant further reduces the number of queries in practice. Our results show that observable validity reactions form a practical key-recovery channel in CKKS deployments exposing this surface, even when the underlying cryptographic assumptions remain intact.

1. Introduction

Fully Homomorphic Encryption (FHE) [41] enables computations to be performed directly on encrypted data. In a typical outsourcing scenario, a client encrypts its input, sends the ciphertext to an untrusted server, and later decrypts the returned ciphertext to obtain the result of the computation, without revealing the underlying data to the server.

This capability makes FHE a natural foundation for secure outsourced computation, including privacy-preserving

machine learning and other cloud-based data-processing tasks [3], [29], [52], [53], [57], [64], [71], [73]. Although FHE implementations have grown increasingly practical, the security of these real-world deployments is no longer determined only by the hardness of the underlying cryptographic assumptions. It additionally hinges on how the resulting ciphertexts are returned, checked, and handled by the client.

Attacks on Secure Outsourced Computation. Attacks on secure outsourced computation based on FHE can be broadly divided into two classes. The first class targets the cryptographic primitives themselves, independent of the specific application. This includes the cryptanalysis of the underlying Ring-Learning-With-Errors (RLWE) assumptions, a foundational hardness assumption widely used in lattice-based cryptography, or exploiting the gap in theoretical security notions, most notably, the fact that standard FHE schemes guarantee Indistinguishability under Chosen-Plaintext Attack (IND-CPA) but are inherently vulnerable to Indistinguishability under Chosen-Ciphertext Attack (IND-CCA) attacks.

In contrast, this paper focuses on the second class of attacks, which emerge from the deployment and protocol layers surrounding homomorphic encryption. While much of the applied FHE literature assumes an honest-but-curious server, this assumption is often overly optimistic for practical deployment, where a server might deviate from the protocol. In such scenarios, a malicious evaluator can actively manipulate the computation, submitting adversarially generated ciphertexts to observe how the client reacts—whether they accept, reject, retry, or leak information through decrypted outputs.

Reaction Attacks. This specific vulnerability is exploited by reaction attacks [8], [16], [17], [28], [31], [46], [48], [63], which turn a client’s visible behavior—such as acceptance (accept or reject), rejection (retry or not), or error handling—into a one-bit decryption oracle. Although each individual response leaks only a single bit of information, a sequence of adaptively chosen queries can amplify this leakage into substantial information about the secret key.

To illustrate this setting, depicted in Figure 1, consider a client C that outsources a computation Φ to a server S , typically a cloud service provider. The client encrypts a message m under the Cheon–Kim–Kim–Song (CKKS)

*. Authors are listed in alphabetical order; see <https://www.ams.org/profession/leaders/culture/JointResearchandItsPublicationfinal.pdf>.

[†]. Corresponding author.

scheme [26], an FHE scheme specialized for real and complex number vectors, sends the ciphertext $\text{Enc}(m)$ to S , and expects to receive an evaluated ciphertext decrypting to an approximation of $\Phi(m)$. In an idealized model, the client simply decrypts the returned ciphertext to obtain the result. In practice, however, a client *seldom* accepts the decrypted output *unconditionally*. In practice, deployments (whether or not they use FHE) often perform application-layer admissibility checks (e.g., range or magnitude tests) before consuming a result. This kind of input validation is a well-established defensive-programming technique [68], and reaction attacks beginning with Bleichenbacher [8] and continuing thereafter have repeatedly demonstrated that the resulting accept/reject behavior can leak secret information. For instance, a machine-learning pipeline might verify whether the decoded class logits, the pre-softmax scores, fall within a valid range, or a protocol might check for arithmetic overflow. Whenever the client’s application-level reaction triggered by a failed check (e.g., an implicit retry or an explicit abort) becomes observable to the server, this check effectively transforms into a reaction oracle.

The crucial point is that such an oracle does not reveal plaintexts, nor does it require the adversary to break the underlying RLWE assumption or the IND-CPA security of CKKS. It reveals strictly whether a submitted ciphertext satisfies the client’s validity predicate. In this scenario, a malicious server S can submit a manipulated ciphertext (a, b) and observes only whether the client-side predicate accepts or rejects the decoded output. For example, if C proceeds without requesting a re-computation, the adversary learns that the predicate was accepted; if C retries or raises an error, the adversary learns it failed. Thus, the validity reaction acts as a one-bit decision oracle on the decoded slot magnitudes. Yet, as we demonstrate, this single bit is sufficient to fully recover the CKKS secret key under a natural deployment surface.

Even such a restricted oracle can leak the secret key. To see this, let s be the client’s secret key, where a valid ciphertext (a, b) satisfies the standard decryption relation $b - as = m + e \bmod q$ for a plaintext m , a small encryption noise e , and a ciphertext modulus q . Suppose a malicious server S attempts to guess the secret key by sampling a candidate ring element u and a sufficiently large public perturbation t . The server then sends the modified ciphertext $(a' = a + t, b' = b + ut)$ to C .

If the guess is correct (i.e., $u = s$), the decryption relation remains unchanged since $b' - a's = e \bmod q$, meaning the modified ciphertext stays within the admissible range. If the guess is incorrect (i.e., $u \neq s$), the decryption residual contains the additional term $(t(u - s) + e \bmod q)$, which exceeds the admissibility threshold with high probability due to the magnitude of t . A reaction oracle can therefore distinguish whether a guessed ring element matches the secret key.

While this simple example demonstrates why validity reactions leak information about s , it is exponentially inefficient, equivalent to a brute-force search over a secret space (e.g., of size 3^N for typical uniform ternary secrets, where

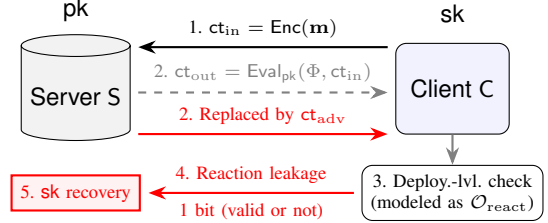


Figure 1: Conceptual Sketch of the Reaction Attack Scenario.

N is the ring dimension).

Crucially, reaction attacks are not captured by standard security notions for FHE, such as IND-CPA, IND-CCA, or the intermediate IND-CPA^D [60], [61], which allows restricted decryption queries. The adversary never obtains the plaintexts themselves, nor does it violate the hardness of the underlying RLWE problem. It merely observes the binary validity outcomes induced by the client’s application-level correctness checks.

1.1. Limitations of Prior Reaction Attacks

To the best of our knowledge, only a few works have explicitly considered reaction attacks on FHE schemes within the secure outsourced computation model [16], [17], [54], [63], [76]. These works have played an important role in showing that client-visible validity responses can become a key-recovery channel. However, they have primarily focused on exact FHE schemes, such as Gentry-type integer schemes [41], [76], TFHE [27], FHEW [34], BGV [14], and BFV [13], [35]; extending them to the approximate CKKS setting is the gap we address.

While crucial in establishing the threat of reaction-based key recovery, these prior attacks were designed for exact schemes and do not transfer directly to RLWE-based approximate FHE such as CKKS [26], where the noisy decoding blurs the deterministic error separation they exploit. The practical difference most relevant to our setting is how queries are formed, discussed below.

Detectability via Ciphertext Reuse. Earlier reaction attacks [16], [17], [63] heavily rely on repeatedly submitting minor variations of a fixed target ciphertext to the client. For instance, in attacks on TFHE/BGV, the adversary fixes a component of the ciphertext (a, b) and issues numerous queries on nearby variants $(a, b + \delta)$. By incrementally adjusting the perturbation δ via a binary-search-style procedure, the adversary aims to pinpoint the exact threshold that triggers a change between a valid and invalid response, thereby exposing the underlying noise. This methodology creates linkable correlations across queries [54]: a client that inspects its access logs may notice many requests in which one ciphertext component is held fixed while the other is systematically varied. Our attack instead re-randomizes every query (Section 3.1), so it does not rely on such a reuse pattern.

These fundamental limitations raise a concrete and challenging question:

Can single-bit validity responses enable efficient secret-key recovery in FHEs using only format-compliant ciphertexts without ciphertext reuse?

1.2. Our Result

We show that a single-bit validity reaction suffices to recover a CKKS secret key [22], [23], [26] without reusing a fixed target ciphertext.

Contributions. Our contributions are four-fold. (i) *Optimality*: full ternary key recovery in $N + |\mathcal{S}| \leq 2N$ deterministic queries, within a factor 1.05–1.26 of the entropy lower bound (Theorem 3.2), and $O(h \log(N/h))$ for sparse secrets. (ii) *A single-bit oracle*: each query returns only one validity bit per ciphertext, not per slot. (iii) *Reuse-resistant queries*: every query is freshly re-randomized and can even be added to an honest circuit output. (iv) *Approximate arithmetic*: the first realization for CKKS, via a public C2S localization and a slot-margin noise analysis (Theorem 3.1).

We state our result informally in Theorem 1.1, with formal versions in Theorem 3.1 (uniform ternary), Theorem 3.5 (sparse ternary), and the matching lower bound in Theorem 3.2.

Public Material Assumed by the Attack. The attack requires only standard public evaluation material: the public encryption key pk and the public rotation keys needed to homomorphically evaluate a Coefficient-to-Slot (C2S) transform. These rotation keys are standard CKKS evaluation keys implementing ring automorphisms, frequently referred to as Galois keys. While widely deployed for CKKS bootstrapping [12], [22], [45], the attack does not strictly require a typical bootstrapping setup. For instance, the attack does not depend on a fixed, pre-shipped key bundle: a reduced key set or on-the-fly key generation [25], [58] similarly suffices. Furthermore, C2S can be evaluated even in non-bootstrappable settings using keys at lower moduli, as in homomorphic matrix computations [6], [70] where only a few Galois keys are provided.²

Theorem 1.1 (Informal Result). *Let N be the ring dimension. For the CKKS scheme with a ternary secret key s , public encryption key pk , and public rotation keys for the Coefficient-to-Slot (C2S) transform, the secret key s can be fully recovered in time polynomial in N using at most $2N$ reaction-oracle queries.*

Furthermore, if the secret key s is restricted to a sparse ternary distribution with h non-zero coefficients, then the secret key s can be fully recovered in time polynomial in N and h , using $O(h \log(N/h))$ reaction-oracle queries.

1. Parameters for Lattigo are described at <https://pkg.go.dev/github.com/Idsec/lattigo/v2/ckks>.

2. as few as 2–3 keys are provided in [6], [70], including a single-slot rotation and conjugation, sufficient to generate the full automorphism group needed for C2S.

We note that in the sparse secret setting, which yields a Shannon entropy of $H(s) = O(h \log(N/h))$ for $h \ll N$.³ As a result, our query complexity asymptotically attains the information-theoretic lower bound, yielding optimal query complexity.

Overview of the Attack. We implemented the full end-to-end attack in Lattigo v6 [56] across various standard parameter regimes ranging from $\log_2 N = 12$ to 16, successfully recovering the secret key with a 100% success rate in all evaluated cases (see Section 4). These ring dimensions and level-zero moduli (also known as bottom moduli q_0) strictly match standard CKKS parameters. Table 1 summarizes the theoretical query-count bounds and the empirical query-counts, with the across the parameter sets, demonstrating full consistency with recent security recommendations [11] and commonly used library presets [56].

Crucially, our evaluation demonstrates that the required oracle calls scale strictly with the structural entropy of the secret key distribution. For standard uniform ternary secrets, full key recovery is consistently achieved within $1.7N$ queries: exactly N queries are utilized to recover the support $\mathcal{S} = \{j : s_j \neq 0\}$, and $h = |\mathcal{S}|$ queries to determine the exact signs.

More devastatingly, under standard sparse-ternary-secret regimes (e.g., fixed-weights of $h \in \{32, 64, 128, 192, 256\}$), our attack employs a bisection-based intersection search to perfectly recover the support in only $O(h \log_2(N/h))$ queries. Once the support is identified, the attack exploits the underlying LWE relation to map the remaining sign-recovery task into a low-dimensional LWE instance. This allows us to bypass further oracle queries entirely and solve the instance via standard lattice reduction (LLL), in a matter of seconds to minutes (e.g., 0.4 seconds for $h = 32$, and 3.3 seconds for $h = 64$). According to the lattice estimator [2], this dimensional collapse reduces the theoretical hardness to only 33–43 bits for $h = 32$ –1024, which is well below the standard 128-bit security.

Artifact. Our artifacts are available at <https://github.com/JiseungKim90/reaction-attack>. The released artifact comprises the attack codes, and the parameter-sweep scripts, built under Go 1.24 against Lattigo [56].

Scope of the Attack. Our attack is protocol-level rather than primitive-level, exploiting correctness validation leakage without breaking the underlying RLWE or CKKS IND-CPA security. It applies to deployments that expose validity-dependent responses to the evaluator. Because queries are independently randomized with fresh encryptions, the attack inherently evades simple ciphertext-reuse filters. We detail the exact scope in Section 3.1, and discuss countermeasures in Section 5.

Distinguishing from Recent IND-CPA^D Attacks. A closely related line of work studies oracle-based key recovery in the IND-CPA^D setting [7], [21], [44], [60]. We stress

3. Since $|\text{key space}| = \binom{N}{h} \cdot 2^h \leq (2eN/h)^h$ holds for $1 \leq h \leq N$.

	Uniform Ternary Secret (Algorithm 1)				Sparse Ternary Secret (Algorithm 2)			
Ref.	$(\log N, \log q_0, \log \Delta)$	Avg. $\#Q$	$\#Q/H(s)$	E2E Time	$(\log N, h)$	Avg. $\#Q$	$\#Q/H(s)$	LLL Time
Lattigo [56] ¹	(12, 35, 30)	6,832	1.052	30 s	(12, 32)	338	1.134	0.4 s
	(12, 37, 32)	6,826	1.051	30 s	(12, 64)	598	1.117	3.3 s
	(14, 45, 40)	27,313	1.052	506 s	(12, 128)	1,079	1.142	1,521 s
	(16, 55, 50)	109,232	1.052	8,529 s	(14, 128)	1,469	1.221	1,521 s**
Guidelines [11]	(16, 60, 58)	109,219	1.051	68,408 s*	(16, 128)	1,801	1.234	1,521 s**

TABLE 1: Overview of Empirical End-to-End Attack (achieving a 100% success rate across all evaluated cases). The baseline attack (Algorithm 1) reports average query-count that tracks closely the theoretical $N + h \approx 5N/3$ for uniform ternary secrets. End-to-end attack wall time is reported in seconds. The variant targeting sparse ternary secrets (Algorithm 2) of fixed Hamming weights ($h \ll N$) compresses the required oracle queries; then the LLL-based sign-recovery completes the secret key recovery, whose cost is reported in seconds. $\#Q$ denotes the number of queries, whose theoretical lower bound is Shannon entropy $H(s)$. More details can be found in Section 4.

that our attack is *not* an IND-CPA^D attack; rather, the two settings are *incomparable*, differing along two axes.

First, the *adversarial class* differs. IND-CPA^D security constrains the adversary to *legitimate* ciphertexts: outputs of the scheme’s own encryption or honest evaluation, tracked together with their underlying plaintexts. Our attacker is a malicious evaluator that submits an *arbitrary*, adversarially crafted ciphertext lying outside this legitimate evaluation tree; a deployment that enforces such legitimacy is therefore *not* in our scope.

Second, the *oracle interface* differs. The IND-CPA^D attack on bootstrapping in [21] also uses only the correctness information, similar to our setting, but for *every message slot*. On the other hand, our attack observes only a single validity bit per query, making it harder to pinpoint which slot caused the failure.

Consequently, the two lines of work address different deployment surfaces and leakage channels, and should be viewed as distinct and potentially cumulative threat models.

1.3. Related Work

IND-CVA Security. Not specifically in the FHE context, however, Bleichenbacher [8] and Hall, Goldberg, and Schneier [46] introduced classical reaction attacks on public-key systems. Hu, Sun, and Jiang [48] later formalized such attacks through indistinguishability under a chosen-ciphertext verification attack (IND-CVA), where the adversary receives a one-bit validity response for adaptively submitted ciphertexts. IND-CVA comes in a non-adaptive and an adaptive form; our attack is adaptive, since its sign-recovery phase (and the sparse-secret bisection) chooses each query as a function of previous validity bits. This abstraction captures the kind of verification signal used in our attack. Loftus et al. [63] showed that FHE can remain vulnerable to IND-CVA attacks even when it satisfies IND-CCA1 security; that is, achieving the stronger CCA1 notion does not by itself close the verification channel. Das, Dutta, and Adhikari [31] further positioned IND-CVA between IND-CPA and IND-CCA security and related it to models such as IND-CCA1.5.

Comparison with Prior Reaction Attacks on FHE. We summarize how our attack relates to prior reaction and reaction-style key-recovery attacks on FHE. Zhang, Plantard, and Susilo [76] gave, to our knowledge, the earliest reaction attack on FHE-based outsourced computation, recovering the secret key of Gentry’s scheme and of integer-based FHE. Their attack is enabled by the bootstrapping key of Gentry’s framework, which publishes an encryption of each secret-key bit $\{\text{Enc}(s_i)\}$; a validity reaction on the i -th such ciphertext then directly leaks s_i , and they observe that its influence on the failure rate is small enough to remain inconspicuous. Their reaction signal is probabilistic (a timing distribution D_0 vs. D_1) and their success requires repeated queries on a fixed message to build confidence. Our setting is fundamentally different: CKKS (like BGV/BFV) publishes no encryption of the secret key, so the coordinate-wise leakage that [76] obtains from the bootstrapping key is simply unavailable, an obstruction also noted by [17, p. 462]. We instead *construct* a secret-bearing ciphertext from the public encryption key via a shift and a public C2S transform, and each query yields a deterministic validity bit rather than a statistical one.

Loftus et al. [63] exhibit an IND-CVA attack on a CCA1-secure SHE scheme, using coefficient-wise verification queries on exact ciphertexts.

Chaturvedi et al. [16], [17] recover keys of the exact schemes TFHE/FHEW and BGV/BFV by repeatedly submitting perturbed variants of a target ciphertext and using the resulting decryption failures as the reaction signal; the BGV/BFV variants inject the perturbation after a modulus switch so that it affects the error rather than the message [17, p. 470]. Their recovery is probabilistic and, by their own classification, non-adaptive (IND-CVA1): each secret coefficient is recovered by guessing its value and the error sign and submitting several samples per guess, so the query count carries a repetition factor (e.g., $\sim 2rN$ for their $r = 11$), whereas ours is deterministic, adaptive, and query-optimal ($N + |S|$, or $O(h \log(N/h))$ for sparse secrets).

We note that the *existence* of such a break is unsurprising: Walter [74] observes that a ciphertext-verification oracle yields a CCA2-style attack that is inherently devastating to

any IND-CPA-secure LWE encryption, and that for exact schemes it can be mounted without side-channel assistance. Our contribution is therefore not the existence of a reaction break but its optimal query complexity and its realization for the *approximate* CKKS scheme, where the per-coordinate noise-extraction underlying such exact-scheme attacks does not directly apply.

Kumazaki and Kaji [54] study the minimum number of oracle queries for TFHE reaction attacks and give a nearly optimal scheme; their optimality is specific to the LWE/TFHE reaction model, whereas our optimality is measured against the Shannon entropy of the ternary CKKS secret.

Finally, the ciphertext-crafting step behind our public-key shift is algebraically close to the plaintext-recovery technique of Chenal and Tang [20]; the distinction is that [20] assumes a decryption oracle returning exact plaintext bits on exact schemes, whereas we operate under a strictly weaker aggregate reaction bit, localize CKKS coefficients through C2S in approximate arithmetic, and re-randomize each query.

Other Security Analyses of FHE Schemes. Most security analyses of CKKS-like schemes consider different oracle surfaces. One line of work studies the IND-CPA^D setting [18], [21], [44], [60], [61], in which the adversary is restricted to *legitimate* ciphertexts (honest encryptions and evaluations tracked with their plaintexts). Some of these attacks, notably [21], already rely on a correctness bit rather than on recovered plaintexts; the distinction from our work is thus not the amount of leakage per query but the adversarial class (legitimate-only versus an arbitrarily crafted ciphertext) and the per-slot versus aggregate granularity (Section 1.2). Hwang et al. [50] study a differentially private homomorphic-evaluation notion for protecting the sender’s message against a malicious receiver in CKKS-based protocols.

Another line of work studies implementation-level leakage, including side-channel and fault-injection attacks [5], [28], [42], [66]. These attacks rely on physical leakage, implementation behavior, or fault effects, whereas our adversary uses only public evaluation material and one observable decoded-slot validity bit.

This is the first polynomial-query reaction attack that recovers the full CKKS secret key from a decoded-slot validity oracle in deployments exposing a public C2S transform. The attack therefore fills a gap between IND-CPA^D analyses, which grant the adversary decryption only on legitimate, evaluation-consistent ciphertexts, and implementation attacks, which require physical or implementation-specific leakage.

2. Preliminaries

Let $\mathbb{Z}$ be the set of all integers. For any integer $q \geq 2$, we define $\mathbb{Z}_q = [-q/2, q/2) \cap \mathbb{Z}$, where $z \bmod q$ denotes the reduction of an integer z into this interval. Unless specified otherwise, all logarithms are assumed to be base 2.

For any set S , sampling x from a uniform distribution over S is denoted by $x \leftarrow S$. When x is drawn from a distribution D , it is written as $x \leftarrow D$. For any positive integer n , the set $\{0, \dots, n-1\}$ is denoted by $[n]$. For any $u \in \mathbb{Z}$, the rounding operation is denoted by $\lfloor u \rfloor$. If $\mathbf{r}$ is a polynomial or a coefficient vector, $\lfloor \mathbf{r} \rfloor$ is applied coefficient-wise. For a subset $S \subseteq \{0, \dots, N-1\}$, let $\mathbf{a}|_S \in \mathbb{R}^{|S|}$ denote the restriction of a vector $\mathbf{a} \in \mathbb{R}^N$ to the coordinates in S . For a matrix $A \in \mathbb{R}^{M \times N}$, $\mathbf{A}|_S \in \mathbb{R}^{M \times |S|}$ denotes the submatrix consisting of the columns of $\mathbf{A}$ indexed by S .

Let $\mathcal{R} = \mathbb{Z}[x]/\langle x^N + 1 \rangle$, where N is a power of two integer. For any integer $q \geq 2$, let $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. For a polynomial $\mathbf{a} = \sum_{j=0}^{N-1} a_j x^j \in \mathcal{R}$, we write its coefficient embedding as $\text{coeff}(\mathbf{a}) := (a_0, \dots, a_{N-1}) \in \mathbb{Z}^N$. When clear from the context, we abuse notation by using bold-face letters to denote ring elements or, equivalently, their corresponding coefficient vectors. We define the ℓ_∞ -, ℓ_1 -, and ℓ_2 -norms of a polynomial $\mathbf{a}$ with its coefficients, by $\|\mathbf{a}\|_\infty := \max_{0 \leq j \leq N-1} |a_j|$, $\|\mathbf{a}\|_1 := \sum_{j=0}^{N-1} |a_j|$, and $\|\mathbf{a}\|_2 := \sqrt{\sum_{j=0}^{N-1} a_j^2}$, respectively. Throughout, an unsuperscripted norm $\|\cdot\|$ refers to this coefficient embedding; the superscripted norm $\|\mathbf{p}\|_\infty^{\text{slot}} := \|\text{Decode}_\Delta(\mathbf{p})\|_\infty$ denotes the ℓ_∞ -norm of the decoded CKKS slot vector (or, slot-norm) with a scale factor Δ , where Decode_Δ and the C2S transform are detailed in Section 2.3.

Let $\mathcal{R}$ and q be as defined above, and let χ be an error distribution over $\mathcal{R}$. For a secret $\mathbf{s} \in \mathcal{R}_q$, a sample from the Ring-Learning-With-Errors (RLWE) distribution with respect to q , $\mathbf{s}$, and χ is generated by sampling $\mathbf{a} \leftarrow \mathcal{R}_q$ uniformly at random and $\mathbf{e} \leftarrow \chi$, and outputting the pair $(\mathbf{a}, \mathbf{b} = \mathbf{a} \cdot \mathbf{s} + \mathbf{e} \bmod q) \in \mathcal{R}_q^2$. The error distribution χ is typically chosen as a zero-centered discrete Gaussian distribution over $\mathcal{R}$, where every coefficient is sampled independently from a discrete Gaussian distribution over $\mathbb{Z}$, denoted by $\chi_{\mathbb{Z}}$. Common choices for the secret distribution include a zero-centered discrete Gaussian, or narrower distributions such as the uniform ternary or fixed-weight ternary distributions.

2.1. Fully Homomorphic Encryption (FHE)

A fully homomorphic encryption (FHE) consists of four procedures.

- $\text{Setup}(1^\lambda)$: For the security parameter λ , the algorithm returns a tuple of keys $(\text{pk}, \text{sk}, \text{evk})$.
- $\text{Enc}(\text{pk}, m)$: For the public key pk and a message m , the algorithm returns a ciphertext ct .
- $\text{Dec}(\text{sk}, \text{ct})$: For the (master) secret key sk and a ciphertext ct , it outputs a value m' .
- $\text{Eval}(\text{pk}, \text{evk}, \mathcal{C}, \text{ct}_1, \dots, \text{ct}_k)$: For the evaluation key evk , a circuit $\mathcal{C}$ and a bundle of ciphertext $\{\text{ct}_i\}_{i=1}^k$, it returns another ciphertext ct' .

We say FHE is correct if for every pair (pk, sk) output by Setup , for any bounded depth circuit $\mathcal{C}$, and any message

$m_1, \dots, m_k$, the following holds with overwhelming probability:

$$\text{Dec}(\text{sk}, \text{Eval}(\text{pk}, \mathcal{C}, \text{ct}_1, \dots, \text{ct}_k)) = C(m_1, \dots, m_k),$$

where $\text{ct}_i \leftarrow \text{Enc}(\text{pk}, m_i)$ for all $i \leq k$.

2.2. Ciphertext Format of CKKS Scheme

This section briefly reviews the ciphertext format for the CKKS FHE scheme [26]. Understanding this format is sufficient to follow our attack. For any RLWE-based FHE scheme, every ciphertext is a pair of ring elements of the form $(a, b) \in \mathcal{R}_q^2$. Let s be the secret polynomial in $\mathcal{R}_q$ with small coefficients.

Given the primitive $2N$ -th root of unity ζ , define $\zeta_i = \zeta^{5^i}$ for an index i . Then, the discrete Fourier Transformation map $\text{DFT}_N : \mathbb{R}[x]/(x^N + 1) \rightarrow \mathbb{C}^{N/2}$ defined by $m(x) \mapsto (m(\zeta_0), \dots, m(\zeta_{N/2-1}))$ is an isomorphism, with its inverse iDFT_N . Based on these maps, we introduce the encoding and decoding algorithms used in the CKKS FHE scheme:

- **Encode** $(\tilde{m}, \Delta)$: Given $\tilde{m} \in \mathbb{C}^{N/2}$ and a public scaling factor Δ , it outputs $\mathbf{m} = \lfloor \Delta \cdot \text{iDFT}_N(\tilde{m}) \rfloor \in \mathcal{R}$.
- **Decode** $(\mathbf{m}, \Delta)$: Given $\mathbf{m} \in \mathcal{R}$ and a public scaling factor Δ , it outputs $\tilde{m} = \text{DFT}_N(\Delta^{-1} \cdot \mathbf{m}) \in \mathbb{C}^{N/2}$.

We refer to the resulting encoded message $\mathbf{m} \in \mathcal{R}$ as the plaintext. A CKKS ciphertext corresponding to the plaintext $\mathbf{m}$ satisfies $b - a \cdot s = \mathbf{m} + e \bmod q$, where the noise polynomial e is initially sampled from a discrete Gaussian distribution with a standard deviation 3.19, and is subsequently updated by homomorphic operations.

Let $\text{ct} = (a, b)$ be a ciphertext and $\text{sk} = (-s, 1)$ be a secret key of CKKS.⁴ For the decryption correctness, CKKS libraries [30], [47], [56], [69], [72] configure parameters such that

$$\begin{aligned} \|\langle \text{ct}, \text{sk} \rangle \bmod q\|_\infty &= \|b - a \cdot s \bmod q\|_\infty \\ &= \|\mathbf{m} + e\|_\infty \leq \Delta. \end{aligned} \quad (1)$$

for the (pre-determined) scaling parameter Δ . This requires the condition $\Delta < q$ to hold, ensuring the scaled message can be accurately recovered. Libraries [1], [56] often configure parameters where $\log q \approx \log \Delta + 10$, providing a sufficient margin between Δ and q .

Secret Distributions. In recent FHE schemes [11], [14], [26], [34], [35] and libraries [1], [30], [47], [56], [72], the secret key s is usually drawn from a set of ternary polynomials $\mathcal{T}^N[x] = \{\sum_{i=0}^{N-1} a_i x^i : a_i \in \{-1, 0, 1\}\}$. It has been used to improve the bootstrapping algorithm of FHE schemes, one of the main bottlenecks of the practicality of FHE applications.

Another (well-known) choice of s is a discrete Gaussian with standard deviation $\sigma = 3.2$ [1], [47], [56], for which

4. For the sake of simplicity, we do not strictly distinguish between the secret polynomial s and the tuple sk , referring to both as the secret key.

$\|s\|_\infty \leq 6\sigma$ in the coefficient embedding with high probability at the concrete dimensions we test; asymptotically, a union bound over the N coefficients gives $\|s\|_\infty = O(\sigma\sqrt{\log N})$ with overwhelming probability (concretely $S = \sigma\sqrt{2\ln(2N/\epsilon)}$ for failure probability ϵ), matching the error bound used above. The quantity relevant to the per-index recovery of Section 3 is this coefficient ℓ_∞ bound, so we take $S = 1$ for ternary secrets and $S = 6\sigma$ for Gaussian secrets, matching the values used in our experiments.

2.3. Public Keys and Coefficient-to-Slot Transform

Beyond the public encryption key $\text{pk} = (a, b = as + e)$, CKKS deployments often release *rotation keys*. The rotation keys $\{\text{rotk}_g\}_{g \in \Gamma}$, also called *Galois keys*, are used to perform homomorphic ring automorphisms $p(x) \mapsto p(x^g)$ on ciphertexts. These automorphisms implement slot rotations, complex conjugation, and the Coefficient-to-Slot (C2S) transform.

We say that a CKKS deployment exposes *C2S-capable public material* if a rotation-key set $\{\text{rotk}_g\}_g$, sufficient to evaluate a C2S layer, is released. This is usually the surface of bootstrapping-enabled or coefficient/slot-conversion workloads, *not* the default of every CKKS deployment; however, it represents the most common choices in secure outsourcing applications.

One such sufficient set is the full automorphism group Γ generated by $x \mapsto x^5$ together with the conjugation automorphism, which is the same type of rotation-key material exposed by Lattigo [56], HEaAN [30], and OpenFHE [1] whenever the workload performs bootstrapping or coefficient/slot conversion. Although C2S is most commonly identified with CKKS bootstrapping [12], [22], [45], the same rotation key set is also released whenever the workload invokes a homomorphic linear transform that requires the conjugation automorphism [45] or any protocol that interconverts coefficient and slot representations. Deployments that release only a strict subset of Γ insufficient to span a single C2S layer fall outside this assumption; a shallow-depth application sharing rotation keys for one fixed linear map is one such case.

Normalized Coefficient-to-Slot Transform. The normalized C2S transform is a public homomorphic linear map on CKKS ciphertexts that maps the coefficient vector of the encrypted polynomial into encrypted slot values. Concretely, given a ciphertext encrypting a polynomial $p = \sum_{j=0}^{N-1} p_j x^j$ at scale Δ , we denote C2S_Δ that returns a pair of ciphertexts $(\text{ct}_R, \text{ct}_I)$ at scale Δ , each with $N/2$ slots. Overloading the decoding notation to ciphertexts, we write $\text{Decode}_\Delta(\text{ct}) := \text{Decode}(\langle \text{ct}, \text{sk} \rangle \bmod q, \Delta)$ for the slot vector obtained by decrypting ct and decoding the resulting polynomial. With this convention,

$$\begin{aligned} \text{Decode}_\Delta(\text{ct}_R) &\approx \Delta^{-1}(p_0, p_1, \dots, p_{N/2-1}) \in \mathbb{R}^{N/2}, \\ \text{Decode}_\Delta(\text{ct}_I) &\approx \Delta^{-1}(p_{N/2}, p_{N/2+1}, \dots, p_{N-1}) \in \mathbb{R}^{N/2}, \end{aligned}$$

up to a bounded additive slot-domain noise e_{C2S} ; our attack requires $\|e_{\text{C2S}}\|_\infty^{\text{slot}}$ to satisfy the slot-margin of The-

orem 3.1, which is then empirically validated in Section 4 for the evaluated Lattigo regimes. In words, the input polynomial’s coefficient vector, viewed at scale Δ , is exposed as two real-valued slot vectors. The construction realizes this as the standard CKKS C2S layer [12], [22], [45] followed by a conjugation-based real/imaginary split using the public conjugation rotation key.⁵ The transform consumes ℓ_{C2S} multiplicative levels, where $\ell_{\text{C2S}} \in \{3, 4\}$ for typical bootstrap-quality decompositions used by [1], [30], [56], and uses only the rotation keys.

We use C2S_Δ as a public primitive: the attack presented in Section 3 invokes C2S_Δ exactly once and treats its slot-domain noise as a constant additive term in the final correctness analysis.

3. Reaction Attack on CKKS and Its Variants

This section presents the threat model and the required definitions, followed by our reaction attack with its correctness proof. We first fix the algebraic setting and secret bound, then define the observable slot-domain reaction oracle, and finally show how the public rotation keys for C2S turn that one bit into coordinate-wise recovery of s . Throughout, we let $\|s\|_\infty \leq S$ in the coefficient embedding (i.e., $\max_j |s_j| \leq S$), for some small integer $S > 0$. This is standard for secret keys in the FHE context, using ternary secrets $S = 1$.

Two Metric Domains. Throughout the attack, the processed ciphertext transitions between two distinct representations: the coefficient domain and the slot domain. The secret bound S , the correctness requirement of Equation (1), and the modulus constraints in Theorem 3.1 are all formulated in the coefficient domain, where the target of recovery is the integer coefficient s_j of the secret key s satisfying $\max_j |s_j| \leq S$ (or holding with overwhelming probability over the choice of the secret key). By contrast, the reaction oracle operates strictly in the slot domain, where validity checks are evaluated via the $\|\cdot\|_\infty^{\text{slot}}$ norm against a threshold τ (see Section 3.2).

The public transformation C2S (see Section 2.3) bridges these two domains, homomorphically relocating the secret key coefficients into corresponding slots. This allows the adversary to isolate a single coefficient and establish a direct operational link to the oracle’s response bit (see Figure 2).

3.1. Threat Model

We consider a secure-outsourcing protocol where an honest client (C) delegates homomorphic computation to a malicious server (S). The client C generates the keys, releases the public material, and applies an application-level admissibility predicate to the decoded slot values of any returned ciphertext before proceeding with the protocol. For each reaction query, the server S receives only a single bit of leakage: whether the queried ciphertext is *valid* or *invalid*.

5. Lattigo’s `dft.RepackImagAsReal` format fuses both steps and returns the pair $(\text{ct}_R, \text{ct}_I)$ directly.

Public Material. The client C publishes the public encryption key $\text{pk} = (a, b = as + e)$ and the set of public rotation keys $\{\text{rotk}_g\}_g$ required to evaluate the application’s homomorphic linear transforms, including the C2S keys when the workload utilizes it. The adversary may exploit only these honestly generated public values; affecting the key generation is strictly prohibited. This directly matches the standard “honest key generation, malicious computation” model established in prior literature [16], [17], [63].

Reaction Oracle. Upon decrypting and decoding a returned ciphertext, the client C checks whether the magnitudes of the decoded slots lie within an admissible range bounded by a threshold τ (see Section 3.2 for a typical instantiation). The resulting validation bit is observable to S either directly (e.g., via an explicit abort signal) or indirectly (e.g., by observing whether the protocol continues or halts). We formally model this verification mechanism as a slot-domain reaction oracle $\mathcal{O}_{\text{react}}$.

Real-World Reaction Leakage. The bit appears whenever the client’s decoded slot vector is checked against an admissible range and the outcome affects an observable protocol step. Concrete examples include (a) outsourced inference workflows where the client decodes class logits, treats a magnitude or range violation as a malformed response, and either re-issues the query or aborts, which the server observes as a retry/no-retry signal; (b) protocols that use the decoded value as the input to a public follow-up computation, so that an obvious overflow leaks via the next-round ciphertext; (c) deployments such as those discussed in [16], [17], [60], [61] that already model an explicit *valid/invalid* response.

Note that while our attack systematically exploits this reaction bit to extract the key, it operates strictly at the protocol layer; it does not compromise the underlying RLWE hardness assumption or violate the IND-CPA security of the CKKS primitive itself.

Adversarial Goal. For given public material and adaptive query access to the reaction oracle $\mathcal{O}_{\text{react}}$, the adversary’s (server’s) goal is to fully recover the secret key s .

Scope of the Attack. The reaction oracle arises under three deployment conditions taken together: (i) the client decrypts and decodes ciphertexts supplied by the evaluator; (ii) it applies an application-level admissibility check (e.g., a magnitude or range test) to the decoded slots; and (iii) the outcome of that check influences some behavior observable to the evaluator, such as a retry, an abort, an error, or a subsequent protocol message. Only when all three hold, the attack applies.

In particular, two natural conditions place a deployment outside our scope. First, if every decryption-dependent behavior is made independent of the admissibility outcome (e.g., a standardized response regardless of the check), condition (iii) fails and no reaction bit is exposed. Second, if the client decrypts only *legitimate* ciphertexts, which is assumed by the IND-CPA^D model, removes the surface. Our threat model targets the common intermediate case in which

arbitrary evaluator-supplied ciphertexts are decrypted and a decryption-dependent behavior is observable.

Since the attack is mounted with fresh ciphertexts for every query, the transcript does not exhibit strong query-to-query correlations, circumventing defenses that flag structured perturbations of a fixed target, though per-key monitoring still applies (Section 5). Consequently, relying on FHE alone is insufficient. Deployments must mitigate these reaction-based side channels, for example, by hardening client responses or limiting oracle access.

We do not consider configurations where the server S directly learns decrypted plaintext values, such as in standard chosen-ciphertext (CCA) or CPA^D attack models. Furthermore, the server S has no control over the key generation process, nor the client’s internal randomness. Traditional physical-layer side channels, including timing analysis, power analysis, and fault-injection attacks, are similarly out of scope.

3.2. Definition of Reaction Oracle

We instantiate the reaction oracle directly in the *decoded-slot* domain, mirroring the practical validation predicates a CKKS client applies post-decoding. The client decodes the returned ciphertext under its designated scale Δ and evaluates whether the resulting slot magnitudes conform to the deployment’s expected range. We do not require the oracle to use any internal awareness or measurement of the underlying CKKS noise; only a simple range check is sufficient.

Let $\tau > 0$ be an admissibility threshold defined by the deployment constraints. The default configuration in our framework is $\tau = 1$, aligning with standard CKKS messages satisfying $\|m\|_{\infty}^{\text{slot}} \leq 1$. Given a queried ciphertext $\text{ct} = (a, b) \in \mathcal{R}_q^2$, we denote the maximum decoded slot magnitude as $\mu(\text{ct})$, defined by:

$$\mu(\text{ct}) := \|b - as \bmod q\|_{\infty}^{\text{slot}} \\ = \|\text{Decode}_{\Delta}(b - as \bmod q)\|_{\infty}.$$

Using this metric, we formally define the *slot-domain reaction oracle* as:

$$\mathcal{O}_{\text{react}}(\text{ct}) = \begin{cases} \text{valid} & \text{if } \mu(\text{ct}) \leq \tau, \\ \text{invalid} & \text{otherwise.} \end{cases}$$

The standard coefficient-domain correctness condition, $\|b - as \bmod q\|_{\infty} \leq \Delta$, is employed across major FHE implementations [1], [56], [72] to guarantee decryption correctness. This coefficient-level bound inherently implies a corresponding slot-domain bound, up to the operator norm of the canonical CKKS decoding map. In this work, however, we take the observable slot-domain predicate itself as the reaction oracle, precisely matching the client’s system-level deployment scenario.

Query Model. The adversary is permitted to adaptively query $\mathcal{O}_{\text{react}}$ with arbitrary ciphertexts under the target key.

Crucially, the submitted ciphertexts are neither reused nor structurally unrealistic.

Instead, each query is independently randomized by adding a fresh or homomorphically evaluated encryption of zero (a construction we detail in Section 3.3). As a result, the adversarial transcript never resubmits an identical ciphertext vector, nor does it present bit-by-bit deterministic modifications of a fixed target ciphertext. Prior to decryption, the query is a syntactically valid CKKS ciphertext, and is computationally indistinguishable from a fresh encryption under the RLWE assumption; hence, transcript-filtering or correlation-based defenses do not apply. However, per-key monitoring of query volume or rejection rate remains a valid detector (Section 5).

3.3. Reaction Attack on CKKS

The attack links each oracle query to a single coefficient of s at a time, using pk and the rotation keys for C2S_{Δ} . Figure 2 sketches the pipeline and Algorithm 1 gives it in full. Note that, for better visibility, k , α , and β are omitted in the superscripts of $\text{ct}^{(3)}$, ct , and ct_{sign} in Algorithm 1.

Offline Step. The adversary first performs two public operations that are independent of the oracle queries. Starting from the public encryption key $\text{pk} = (a, b = as + e)$, it forms $\text{ct}^{(1)} := (a - \Delta, b)$, which decrypts to $\Delta s + e \pmod{q}$.⁶ Since $b - (a - \Delta)s = \Delta s + e \pmod{q}$.

The adversary then applies the public normalized C2S to obtain $(\text{ct}_R^{(2)}, \text{ct}_I^{(2)}) \leftarrow \text{C2S}_{\Delta}(\text{ct}^{(1)})$. This carries the secret key coefficients into slots. Concretely, when decrypted and decoded, the k -th slot of $\text{ct}_R^{(2)}$ contains s_k , while the k -th slot of $\text{ct}_I^{(2)}$ contains $s_{k+N/2}$, each within a noise of slot-norm $\ll \tau$.

Online Step. In the online step, the adversary queries the ciphertexts obtained based on $\text{ct}_R^{(2)}$ and $\text{ct}_I^{(2)}$. We divide the queries into two types: (i) Support queries, which are employed to identify the support set $\mathcal{S} := \{j : s_j \neq 0\}$, and (ii) Sign queries, which are used to determine the sign associated with each index in $\mathcal{S}$.

Phase 1: Support query. A support query tests a single secret coefficient by means of a one-hot plaintext mask. Let $e_k \in \mathbb{R}^{N/2}$ denote the k -th standard slot basis vector. For an amplitude $\alpha > 0$, define the one-hot plaintext mask

$$\pi^{(k, \alpha)} := \text{Encode}(\alpha e_k, \Delta),$$

namely the CKKS plaintext encoding, at scale Δ , of the slot vector with value α in slot k and zero elsewhere.

For $\beta \in \{R, I\}$, define the target secret coefficient by

$$s_{k, \beta} := \begin{cases} s_k, & \text{if } \beta = R, \\ s_{k+N/2}, & \text{if } \beta = I. \end{cases}$$

6. One may need to lower the level if pk ’s modulus is larger than the usual ciphertext modulus, typically via level down or modulus switching.

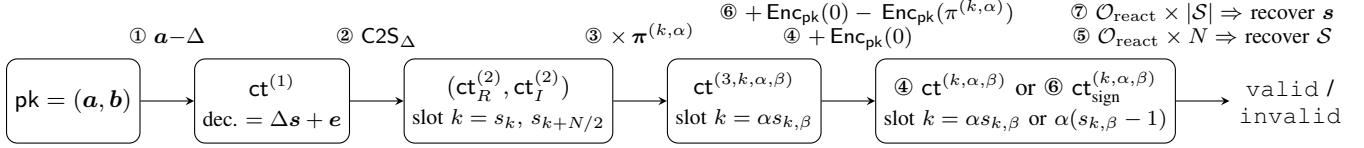


Figure 2: Overview of the Reaction Attack (Algorithm 1). For $0 \leq k < N/2$, $s_{k,\beta}$ denotes s_k or $s_{k+N/2}$, depending on $\beta \in \{R, I\}$. The attack proceeds as follows: **(i) Offline** (①, ②): ① Public-key shift, and ② homomorphic C2S locates a secret key coefficient into a target slots. **(ii) Online Phase 1** (③, ④, ⑤): Support recovery in N queries. ③ One-hot mask vector isolates a target slot, and ④ a fresh encryption re-randomizes it. ⑤ Resulting query recovers the support $\mathcal{S} := \{j : s_j \neq 0\}$. **(iii) Online Phase 2** (③, ⑥, ⑦): Sign recovery in $|\mathcal{S}|$ queries. ③ Isolation, ⑥ re-randomization, and subtraction by an encryption of the mask, deviates the slot values, which ⑦ reveals the sign when queried. The full secret s is then recovered.

The adversary computes

$$\text{ct}^{(3,k,\alpha,\beta)} := \text{ct}_{\beta}^{(2)} \cdot \pi^{(k,\alpha)},$$

which contains a signal only in the k -th slot. The signal is $\alpha s_{k,\beta}$, up to bounded slot-domain noise. The adversary then re-randomizes the query by adding a fresh encryption of zero (at the same level and scale):

$$\text{ct}^{(k,\alpha,\beta)} := \text{ct}^{(0)} + \text{ct}^{(3,k,\alpha,\beta)}, \quad \text{ct}^{(0)} \leftarrow \text{Enc}_{\text{pk}}(0).$$

This fresh zero encryption does not change the decoded signal, but makes the submitted ciphertext independent of previous queries. Therefore, the underlying slot vector has the form

$$\alpha s_{k,\beta} \mathbf{e}_k + \tilde{\delta},$$

where $\tilde{\delta}$ denotes the resulting slot-domain noise. That is, the slot vector effectively constitutes a noisy one-hot vector, carrying $\alpha s_{k,\beta}$ in its k -th coordinate.

We now fix an amplitude $\alpha > 0$ for the attack. This value is chosen so that the noise-only case remains below the reaction threshold τ , whereas the signal produced by any nonzero ternary coefficient exceeds τ . The precise slot-margin and modulus conditions ensuring this separation are stated in the correctness analysis below.

The reaction oracle then distinguishes whether the targeted coefficient is zero:

$$\mathcal{O}_{\text{react}}(\text{ct}^{(k,\alpha,\beta)}) = \begin{cases} \text{valid}, & \text{if } s_{k,\beta} = 0, \\ \text{invalid}, & \text{if } s_{k,\beta} \neq 0. \end{cases}$$

Thus, one oracle query determines whether the coefficient selected by (k, β) lies in the support $\mathcal{S}$. Repeating this for all $k \in \{0, \dots, N/2 - 1\}$ and $\beta \in \{R, I\}$ recovers $\mathcal{S} = \{j : s_j \neq 0\}$ using exactly N reaction queries.

Phase 2: Sign query. A second pass over $\mathcal{S}$ resolves the signs. For $j \in \mathcal{S}$, choose (k, β) so that $s_{k,\beta} = s_j$. The adversary first forms a fresh support-type query ciphertext $\text{ct}^{(k,\alpha,\beta)}$ as above, then subtracts an encryption of the one-hot mask:

$$\text{ct}_{\text{sign}}^{(k,\alpha,\beta)} := \text{ct}^{(k,\alpha,\beta)} - \text{ct}_{\text{off}}^{(k,\alpha)},$$

where $\text{ct}_{\text{off}}^{(k,\alpha)} \leftarrow \text{Enc}_{\text{pk}}(\pi^{(k,\alpha)})$. Then the underlying target slot has signal

$$\alpha(s_{k,\beta} - 1),$$

up to a bounded slot-domain noise. The signal is 0 when $s_{k,\beta} = +1$, and -2α when $s_{k,\beta} = -1$; hence, the oracle accepts exactly the positive coefficients and rejects the negative ones. Thus the sign of each nonzero coefficient is recovered with one additional query, and the total query count becomes

$$N + |\mathcal{S}| \leq 2N.$$

All per-query multiplications are by plaintext masks, so no relinearization key is required.⁷

Mask Amplitude. The offline and per-query noise terms combine as follows. Phase 1 and Phase 2 queries decode respectively to

$$\alpha s_{k,\beta} \mathbf{e}_k + \tilde{\delta} \quad \text{and} \quad \alpha(s_{k,\beta} - 1) \mathbf{e}_k + \tilde{\delta}'.$$

Write

$$\nu_{\alpha} := B_{\text{shift}}^{\text{coeff}} / \Delta + B_{\text{C2S}}^{\text{slot}},$$

for the slot-domain noise carried through C2S_Δ per unit mask amplitude. Then the noise terms are bounded by

$$\begin{aligned} \|\tilde{\delta}\|_{\infty}^{\text{slot}} &\leq \eta_{\text{sup}} := \alpha \nu_{\alpha} + B_{\text{pmult}}^{\text{slot}} + B_{\text{fresh}}^{\text{slot}}, \\ \|\tilde{\delta}'\|_{\infty}^{\text{slot}} &\leq \eta_{\text{sign}} := \eta_{\text{sup}} + B_{\text{off}}^{\text{slot}}, \end{aligned}$$

where $B_{\text{pmult}}^{\text{slot}}$, $B_{\text{fresh}}^{\text{slot}}$, and $B_{\text{off}}^{\text{slot}}$ are the noise bounds introduced by the plaintext masking, the fresh encryption of zero, and the Phase-2 offset, respectively.

The role of α is to separate zero and nonzero ternary coefficients with respect to the reaction threshold τ . When $s_{k,\beta} = 0$, the Phase-1 query contains only noise; when $s_{k,\beta} \neq 0$, the targeted slot contains a signal of magnitude approximately α . Since the mask scales both the signal $\alpha s_{k,\beta}$ and the inherited C2S noise $\alpha \nu_{\alpha}$, changing α does not improve their ratio. It only places the signal and noise relative to the fixed threshold τ .

Ignoring the smaller additive terms $B_{\text{pmult}}^{\text{slot}}$, $B_{\text{fresh}}^{\text{slot}}$, and $B_{\text{off}}^{\text{slot}}$, Phase 1 separates zero from nonzero coefficients whenever

$$\tau < \alpha < \tau / \nu_{\alpha}.$$

The full slot-margin inequalities, including these additive terms and the Phase-2 sign separation, are stated in Theorem 3.1. This admissible window is independent of N , since

7. In Lattigo, MulRelinNew on a plaintext operand does not use r1k.

Algorithm 1: $O(N)$ Reaction Attack on CKKS.

Input: Slot-domain reaction oracle $\mathcal{O}_{\text{react}}$ with threshold τ , public material ($\text{pk} = (\mathbf{a}, \mathbf{b}), \{\text{rotk}_g\}$), parameters (N, q_0, Δ) , and a constant α satisfying the slot-margin and modulus conditions of Theorem 3.1.

Output: Ternary secret $\mathbf{s} = (s_0, \dots, s_{N-1})$.

```

1  $\text{ct}^{(1)} \leftarrow (\mathbf{a} - \Delta, \mathbf{b})$  // Offline shift
2  $(\text{ct}_R^{(2)}, \text{ct}_I^{(2)}) \leftarrow \text{C2S}_\Delta(\text{ct}^{(1)})$  // Offline C2S
3  $\mathcal{S} \leftarrow \emptyset, \mathbf{s} \leftarrow \mathbf{0}$ 
  /* Phase 1: Support recovery, one query per coefficient. */
4 foreach  $\beta \in \{R, I\}$  do
5   for  $k = 0$  to  $N/2 - 1$  do
6      $j \leftarrow k$  if  $\beta = R$ , else  $k + N/2$ 
7      $\text{ct}^{(3)} \leftarrow \text{ct}_\beta^{(2)} \cdot \pi^{(k, \alpha)}$ 
8     Sample fresh  $\text{ct}^{(0)} \leftarrow \text{Enc}_{\text{pk}}(0)$ 
9      $\text{ct} \leftarrow \text{ct}^{(0)} + \text{ct}^{(3)}$ 
10    if  $\mathcal{O}_{\text{react}}(\text{ct}) = \text{invalid}$  then
11       $\mathcal{S} \leftarrow \mathcal{S} \cup \{j\}$ 
  /* Phase 2: Sign recovery, one query per nonzero index. */
12 foreach  $j \in \mathcal{S}$  do
13    $\beta \leftarrow R$  if  $j < N/2$  else  $I$ 
14    $k \leftarrow j$  if  $\beta = R$  else  $j - N/2$ 
15    $\text{ct}^{(3)} \leftarrow \text{ct}_\beta^{(2)} \cdot \pi^{(k, \alpha)}$ 
16    $\text{ct}_{\text{off}}^{(k, \alpha)} \leftarrow \text{Enc}_{\text{pk}}(\pi^{(k, \alpha)})$ 
17   Sample fresh  $\text{ct}^{(0)} \leftarrow \text{Enc}_{\text{pk}}(0)$ 
18    $\text{ct}_{\text{sign}} \leftarrow \text{ct}^{(0)} + \text{ct}^{(3)} - \text{ct}_{\text{off}}^{(k, \alpha)}$ 
19   if  $\mathcal{O}_{\text{react}}(\text{ct}_{\text{sign}}) = \text{valid}$  then
20      $s_j \leftarrow +1$ 
21   else
22      $s_j \leftarrow -1$ 
23 return  $\mathbf{s}$ 

```

both the slot signal α and the threshold τ are independent of N . In our experimental regimes, $\nu_\alpha \ll 1$ holds with a large margin (see Section 4), so we fix α to 4.

3.4. Correctness and Query Complexity

Theorem 3.1. Consider a CKKS deployment with parameters (N, q_0, Δ) , level-zero modulus q_0 , ternary secret $\mathbf{s} \in \{-1, 0, 1\}^N$, public material $(\text{pk}, \{\text{rotk}_g\})$ sufficient for C2S_Δ , and reaction oracle $\mathcal{O}_{\text{react}}$ with threshold $\tau \geq 1$. Let η_{sup} and $\eta_{\text{sign}} = \eta_{\text{sup}} + B_{\text{off}}^{\text{slot}}$ the noise bounds of the decoded slots of Phase 1 and Phase 2 queries, respectively. Suppose an amplitude $\alpha > 0$ satisfies:

(i) Slot-margin.

$$\begin{aligned} \eta_{\text{sup}} < \tau, \quad \alpha - \eta_{\text{sup}} > \tau, \\ \eta_{\text{sign}} < \tau, \quad 2\alpha - \eta_{\text{sign}} > \tau. \end{aligned} \quad (2)$$

(ii) No modular wrap.

$$2\alpha \kappa_{\text{max}} + B_{\text{coeff}} < q_0/2, \quad (3)$$

where $\kappa_{\text{max}} \leq \Delta/\sqrt{N/2}$ is the largest coefficient-domain magnitude produced by encoding a unit-magnitude slot and B_{coeff} bounds the coefficient-domain non-signal terms.

Then Algorithm 1 recovers $\mathbf{s}$ in at most $2N$ reaction-oracle queries and $\text{poly}(N, \log q)$ time, except with negligible probability.

Proof of Theorem 3.1. Fix a pair (k, β) with $k \in \{0, \dots, N/2 - 1\}$ and $\beta \in \{R, I\}$, and write $s_{k, \beta}$ for the targeted coefficient: $s_{k, R} = s_k$ and $s_{k, I} = s_{k+N/2}$.

Decoded slot values. The construction of Section 3.3 shows that the Phase 1 query ciphertext decodes, in the slot domain, to $\alpha s_{k, \beta} \mathbf{e}_k + \tilde{\delta}$ with $\|\tilde{\delta}\|_\infty^{\text{slot}} \leq \eta_{\text{sup}}$, and the Phase 2 query ciphertext to $\alpha(s_{k, \beta} - 1) \mathbf{e}_k + \tilde{\delta}'$ with $\|\tilde{\delta}'\|_\infty^{\text{slot}} \leq \eta_{\text{sign}}$; the additional $B_{\text{off}}^{\text{slot}}$ in η_{sign} accounts for the fresh encryption of the offset $\pi^{(k, \alpha)}$ subtracted in the Phase-2 step of Algorithm 1.⁸

No modular wrap (Condition (ii)). The decoded values read the level-zero decryption residue through the centered reduction modulo q_0 , which preserves the signal only if no coordinate leaves $[-q_0/2, q_0/2]$. The largest coefficient-domain magnitude is the Phase 2 worst case $s_{k, \beta} = -1$, whose encoded slot vector $\alpha(s_{k, \beta} - 1)\mathbf{e}_k$ has ℓ_∞ -norm $2\alpha\kappa_{\text{max}}$; together with the non-signal terms bounded by B_{coeff} the total is below $q_0/2$ by Equation (3). Hence no coordinate wraps and the slot identities above hold.

Phase 1 (support). If $s_{k, \beta} = 0$ the decoded slot vector equals $\tilde{\delta}$, of norm $\leq \eta_{\text{sup}} < \tau$ by Equation (2), so the oracle accepts. If $s_{k, \beta} \neq 0$ then $|s_{k, \beta}| = 1$, so slot k has magnitude $\geq \alpha - \eta_{\text{sup}} > \tau$ and the oracle rejects. One query at each (k, β) with $k \in [N/2]$, $\beta \in \{R, I\}$, therefore decides whether the addressed coefficient is zero, and the N such queries identify the support $\mathcal{S} = \{j : s_j \neq 0\}$.

Phase 2 (sign). For each $j \in \mathcal{S}$ the targeted slot holds $\alpha(s_{k, \beta} - 1) + \tilde{\delta}'_k$ with $s_{k, \beta} - 1 \in \{0, -2\}$. By Equation (2), $s_{k, \beta} = +1$ gives magnitude $\leq \eta_{\text{sign}} < \tau$ (accept) and $s_{k, \beta} = -1$ gives magnitude $\geq 2\alpha - \eta_{\text{sign}} > \tau$ (reject), so one query per $j \in \mathcal{S}$ recovers its sign.

Query count and running time. Phase 1 issues N queries and Phase 2 issues $|\mathcal{S}| \leq N$, for at most $2N$ in total; each query is assembled from public CKKS operations, so the algorithm runs in $\text{poly}(N, \log q)$ time.

Success probability. The key-generation and evaluation-key randomness is drawn once; conditioned on it, the C2S_Δ and key-switching noise terms entering $B_{\text{C2S}}^{\text{slot}}$ and B_{coeff} meet their stated bounds except with negligible probability. Conditioned on such a key, each query's remaining noise depends only on that query's fresh-encryption randomness and stays within $\eta_{\text{sup}}, \eta_{\text{sign}}$ except with negligible probability; a union bound over the $2N$ queries then makes every oracle

8. If the offset is implemented by plaintext subtraction $\text{ct} - \pi^{(k, \alpha)}$ in place of $\text{ct} - \text{Enc}_{\text{pk}}(\pi^{(k, \alpha)})$, the term $B_{\text{off}}^{\text{slot}}$ vanishes; we use the encrypted offset to keep the submitted ciphertext distribution close to a fresh CKKS ciphertext.

answer correct, so s is recovered except with negligible probability. $\square$

Theorem 3.1 gives $2N$ queries; we close by showing this is essentially optimal for any one-bit reaction oracle.

Theorem 3.2. *In the oracle-only model where the public material is treated as information-theoretically independent of s , any algorithm that recovers every uniform $s \in \{-1, 0, 1\}^N$ from $\mathcal{O}_{\text{react}}$ makes at least $\lceil N \log_2 3 \rceil$ queries in the worst case. For the sparse-ternary distribution with $\Pr[s_i \neq 0] = p$, where nonzero coefficients have uniform signs, the average-case lower bound on the number of binary oracle answers is*

$$H_p(s) = N(p \log_2(2/p) + (1-p) \log_2 \frac{1}{1-p}) .$$

Proof. A transcript of T adaptive binary oracle answers takes at most 2^T distinct values. A zero-error algorithm must induce distinct transcripts on distinct secrets, so $2^T \geq 3^N$ for uniform ternary secrets, giving $T \geq N \log_2 3$ in the worst case. For the Bernoulli-sparse ternary distribution, each coefficient is zero with probability $1-p$, and is $+1$ or -1 with probability $p/2$ each. Shannon’s source-coding bound gives the average-case lower bound $\mathbb{E}[T] \geq H_p(s)$ with H_p defined as above. $\square$

Together with Theorem 3.1, the query complexity of Algorithm 1 remains within a factor of $2/\log_2 3 \approx 1.262$ of this information-theoretic lower bound. Note that this abstraction inherently ignores the computational information that the RLWE public key pk carries regarding s ; thus, we claim near-optimality strictly with respect to the number of one-bit oracle responses, rather than against an unbounded adversary that additionally exploits pk . Concretely, the default Lattigo [56] configuration with a uniform ternary secrets ($p = 2/3$) has $H_p(s) \approx 1.585N$. In this regime, the empirical query count of $\approx 1.667N$ observed in Section 4 is within a factor of 1.05, demonstrating the near-optimality of our attack in practice.

Remark 3.3. *The same idea extends to secrets with coefficients in $\{-S, \dots, S\}$. Instead of testing only whether a coefficient is zero, the adversary can perform interval tests. To test whether s_j lies within distance $w\tau$ of a center c , it uses a one-hot mask of amplitude $\alpha = 1/w$ and subtracts an encrypted one-hot offset encoding c in the targeted slot. The decoded target slot then has the form*

$$(s_j - c)/w + \tilde{\delta} .$$

Under the corresponding slot-margin conditions, the reaction oracle tests whether $|s_j - c| \leq w\tau$. Applying bisection over $\{-S, \dots, S\}$ recovers each coefficient in $\lceil \log_2(2S+1) \rceil$ queries, hence $O(N \log S)$ queries in total.

Remark 3.4 (Composition with an honest output). *The Phase-1 and Phase-2 query ciphertexts may instead be added to a ciphertext obtained from an honest evaluation of a circuit that the secret key owner expects. The only change is that the isolated target slot now reads $m_k + \alpha s_{k,\beta}$ rather than $\alpha s_{k,\beta}$; it stays admissible when $s_{k,\beta} = 0$ and*

overflows otherwise, so the reaction still isolates the target while riding on a legitimate result.

3.5. Sparse Secrets and CKKS Variants

The attack described in Algorithm 1 can be generalized in two directions: 1) we can further reduce the query complexity for CKKS with sparse ternary secrets, and 2) we can target CKKS variants whenever they provide a public, normalized C2S_Δ layer that meets the necessary noise margins.

3.5.1. Sublinear Queries for Sparse Secrets. The baseline $2N$ -query bound allocates one oracle query per coordinate to recover the support $\mathcal{S}$. However, when the secret key is sparse, almost all of its uncertainty lies in the support, which can be optimized with far fewer queries using a combinatorial *Group Testing* strategy. Since the reaction oracle thresholds the maximum slot-norm magnitude, the individual one-hot mask ($\pi^{(k,\alpha)}$) can be naturally generalized to a *subset mask*:

$$\pi^{(T,\alpha)} := \text{Encode}\left(\alpha \sum_{k \in T} \mathbf{e}_k, \Delta\right) ,$$

over any set of slots T .

The resulting reaction query triggers an *invalid* response if and only if at least one coordinate within T intersects with the secret support. Crucially, because the oracle evaluates the maximum magnitude across distinct slots, the individual ± 1 signs of the secret coefficients occupy isolated positions and cannot cancel each other out. Each query, therefore, effectively functions as a single group test, asking “Does the subset T intersect the secret support $\mathcal{S}$?” Consequently, an adaptive bisection procedure can identify all $h = |\mathcal{S}|$ non-zero coordinates using only $O(h \log_2(N/h))$ queries.

Once the support $\mathcal{S}$ is fully recovered, the corresponding signs can be extracted directly from the public key without any further oracle queries. Restricting the decryption relation $\mathbf{b} = \mathbf{a}s + \mathbf{e} \pmod{q}$ to the support $\mathcal{S}$ leaves an over-determined, h -dimensional LWE instance (N samples in $h \ll N$ unknowns), where the short secret is the sign vector $\mathbf{s}|_{\mathcal{S}} \in \{-1, +1\}^h$. The concrete hardness of this reduced instance is evaluated empirically in Section 4.

The full algorithm is specified in Algorithm 2, where the recursive binary-splitting procedure is formalized as `BisecGroupTest` in Algorithm 3. The lattice reduction $\text{LatticeReduction}_q(\mathbf{A}, \mathbf{b})$ finds a short vector $\mathbf{s}$ satisfying $\mathbf{A}\mathbf{s} \approx \mathbf{b} \pmod{q}$. Since the LWE dimension collapses from N to the sparse Hamming weight $h \ll N$ while q remains large, this reduced instance can be solved efficiently using standard lattice reduction algorithms (e.g., LLL or BKZ). Note that an adversary can construct the modular relation from any ct , pk , or rotk ; selecting a key with the largest available modulus $q = PQ$ will make the reduction cost optimal.

Theorem 3.5 then proves the correctness of Algorithm 2.

Algorithm 2: Reaction Attack on Sparse-secret CKKS.

Input: Reaction oracle $\mathcal{O}_{\text{react}}$ with threshold τ , public material $(\text{pk} = (\mathbf{a}, \mathbf{b}), \{\text{rotk}_g\})$, modulus q of pk, parameters (N, q_0, Δ) , and amplitude α satisfying Theorem 3.5.

Output: Sparse ternary secret $\mathbf{s} = (s_0, \dots, s_{N-1})$.

```

1  $\text{ct}^{(1)} \leftarrow (\mathbf{a} - \Delta, \mathbf{b})$  // Offline shift
2  $(\text{ct}_R^{(2)}, \text{ct}_I^{(2)}) \leftarrow \text{C2S}_\Delta(\text{ct}^{(1)})$  // Offline C2S
3  $\mathcal{S} \leftarrow \emptyset, \mathbf{s} \leftarrow \mathbf{0}$ 
  /* Phase 1: Support recovery. */
4 foreach  $\beta \in \{R, I\}$  do
5    $\mathcal{S}_\beta \leftarrow \text{BisecGroupTest}(\lceil \frac{N}{2} - 1 \rceil, \text{ct}_\beta^{(2)})$ 
6    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_\beta$ 
  /* Phase 2: Sign recovery. */
7 Construct  $\mathbf{M}_\mathbf{a} \in \mathbb{Z}_q^{N \times N}$  from  $\mathbf{a}$ 
8  $\mathbf{s}|_\mathcal{S} \leftarrow \text{LatticeReduction}_q(\mathbf{M}_\mathbf{a}|_\mathcal{S}, \mathbf{b})$ 
9  $\mathbf{s}|_{\mathcal{S}^c} \leftarrow \mathbf{0}$ 
10 return  $\mathbf{s}$ 

```

Algorithm 3: Subroutine BisecGroupTest.

Input: Target $T \subseteq [\frac{N}{2} - 1]$, and a ciphertext $\text{ct}^{(2)}$.

Output: Local support indices $\mathcal{S}_{\text{local}}$.

Data: Global Parameters: $\mathcal{O}_{\text{react}}, \text{pk}, \Delta, \alpha$

```

1  $\mathcal{S}_{\text{local}} \leftarrow \emptyset$ 
2  $\pi^{(T, \alpha)} \leftarrow \text{Encode}(\alpha \sum_{k \in T} \mathbf{e}_k, \Delta)$  Sample fresh
    $\text{ct}^{(0)} \leftarrow \text{Enc}_{\text{pk}}(0)$ 
3  $\text{ct} \leftarrow \text{ct}^{(0)} + \text{ct}^{(2)} \cdot \pi^{(T, \alpha)}$ 
4 if  $\mathcal{O}_{\text{react}}(\text{ct}) == \text{invalid}$  then
5   if  $|T| == 1$  then
6      $k \leftarrow$  the single element in  $T$ 
7     if  $\text{ct}^{(2)} == \text{ct}_R^{(2)}$  then
8        $\mathcal{S}_{\text{local}} \leftarrow \mathcal{S}_{\text{local}} \cup \{k\}$ 
9     else
10       $\mathcal{S}_{\text{local}} \leftarrow \mathcal{S}_{\text{local}} \cup \{k + \frac{N}{2}\}$ 
11   else
12     Partition  $T := T_1 \sqcup T_2$ , s.t.  $|T_1| = \lfloor |T|/2 \rfloor$ 
13      $\mathcal{S}_{\text{local}} \leftarrow \mathcal{S}_{\text{local}} \cup \text{BisecGroupTest}(T_1, \text{ct}^{(2)})$ 
14      $\mathcal{S}_{\text{local}} \leftarrow \mathcal{S}_{\text{local}} \cup \text{BisecGroupTest}(T_2, \text{ct}^{(2)})$ 
15 return  $\mathcal{S}_{\text{local}}$ 

```

Theorem 3.5. Let $\mathbf{s}$ be a ternary secret with Hamming weight $h < N$, and assume the Phase-1 slot-margin of Theorem 3.1 together with the multi-slot modulus condition $\alpha \kappa_{\max} h + B_{\text{coeff}} < q_0/2$. Then, the group testing in Algorithm 2 recovers the support $\mathcal{S}$ using $O(h \log_2(N/h))$ reaction queries. The sign vector $\mathbf{s}|_\mathcal{S}$ is subsequently determined by solving the restricted h -dimensional LWE instance, requiring no additional oracle queries.

Proof of Theorem 3.5. Recall the Phase-1 slot-margin condition $\eta_{\text{sup}} < \tau < \alpha - \eta_{\text{sup}}$ from Equation (2).

A subset query for group testing. Fix $\beta \in \{R, I\}$ and a slot

subset $T \subseteq [N/2]$. We construct the subset mask $\pi^{(T, \alpha)}$ and compute $\text{ct} = \text{ct}^{(0)} + \text{ct}_\beta^{(2)} \cdot \pi^{(T, \alpha)}$, where $\text{ct}^{(0)} \leftarrow \text{Enc}_{\text{pk}}(0)$. By the coordinate-wise linearity, the slot vector becomes

$$\alpha \sum_{k \in T} s_{j_k} \mathbf{e}_k + \tilde{\delta} \quad \text{for some } \|\tilde{\delta}\|_\infty^{\text{slot}} \leq \eta_{\text{sup}},$$

where $j_k = k$ for $\beta = R$ and $j_k = k + N/2$ for $\beta = I$. The noise bounds $B_{\text{pmult}}^{\text{slot}}$ and $B_{\text{fresh}}^{\text{slot}}$ scale with α and the ring parameters, remaining invariant to the cardinality $|T|$.

Thus, if $T \cap \mathcal{S} = \emptyset$, then every $s_{j_k} = 0$, bounding the decoded magnitude by $\eta_{\text{sup}} < \tau$ (valid). Else if $T \cap \mathcal{S} \neq \emptyset$, there exists at least one slot $k \in T$ with $s_{j_k} = \pm 1$, yielding a magnitude of at least $\alpha - \eta_{\text{sup}} > \tau$, triggering an invalid response. The query output thus perfectly instantiates the group-testing predicate $\mathbf{1}_{[T \cap \mathcal{S} \neq \emptyset]}$.

For the coefficient-domain modulus constraint, linearity and the triangle inequality bound the ℓ_∞ -norm of the multi-slot signal by $\alpha \kappa_{\max} \cdot |T \cap \mathcal{S}| \leq \alpha \kappa_{\max} h$. Modular wrap-around is securely avoided under the condition $\alpha \kappa_{\max} h + B_{\text{coeff}} < q_0/2$.

Support recovery via repeated group testing. We run group testing on each half, independently using $\text{ct}_\beta^{(2)}$. Let $h_R = |\mathcal{S} \cap [0, M]|$ and $h_I = |\mathcal{S} \cap [M, N]|$, where $h_R + h_I = h$ and $M = N/2$. Hwang's generalized binary-splitting algorithm [32], [49] isolates all h_β non-zero indices within $\leq O(h_\beta \log_2(M/h_\beta))$ adaptive queries, which is order-optimal against the information-theoretic bound $\lceil \log_2 \binom{M}{h_\beta} \rceil$ for $\beta \in \{R, I\}$. By the concavity of $t \mapsto t \log_2(M/t)$, the total query complexity is bounded by:

$$\sum_{\beta \in \{R, I\}} h_\beta \log_2(M/h_\beta) \leq h \log_2(2M/h) = h \log_2(N/h).$$

Hence, the entire support $\mathcal{S}$ is recovered in $O(h \log_2(N/h))$ reaction queries.

Sign extraction via lattice reduction. When represented in coefficient vectors, the public key elements satisfy $\mathbf{b} = \mathbf{M}_\mathbf{a} \mathbf{s} + \mathbf{e} \pmod{q}$, where $\mathbf{M}_\mathbf{a} \in \mathbb{Z}_q^{N \times N}$ is the negacyclic matrix of $\mathbf{a}$. Since $\mathbf{s}$ is supported strictly on $\mathcal{S}$, the relation restricts to the $h = |\mathcal{S}|$ columns of $\mathbf{M}_\mathbf{a}$:

$$\mathbf{b} = \mathbf{M}_\mathbf{a}|_\mathcal{S} \cdot \mathbf{s}|_\mathcal{S} + \mathbf{e} \pmod{q},$$

for $\mathbf{M}_\mathbf{a}|_\mathcal{S} \in \mathbb{Z}_q^{N \times h}$ and $\mathbf{s}|_\mathcal{S} \in \{-1, +1\}^h$. This overdetermined h -dimensional LWE instance can be solved via standard reductions, with no additional oracle access. $\square$

Note that given $\kappa_{\max} \leq \Delta/\sqrt{N/2}$, the multi-slot modulus constraint is highly permissive and easily satisfied in typical sparse secret deployments where $h \ll N$. Furthermore, while Theorem 3.2 establishes a rigid reaction-query lower bound based purely on oracle feedback, this variant actively integrates pk. Consequently, the total query complexity may drop below the $H_p(\mathbf{s})$ reaction-query floor whenever the h -dimensional LWE instance is efficiently solvable.

3.5.2. Conditional Applicability to CKKS Variants. Our attack framework extends beyond plain CKKS to architectural variants such as bootstrappable [12], [22], [45], rescaling-free [51], [65], and discrete-message [24], [39], [62] extensions. Any variant exposing a public, normalized $C2S_\Delta$ layer with slot-wise plaintext multiplication is inherently vulnerable; its precise viability depends on the specific variant’s embedding error, scale management, and decryption noise, as analyzed in Corollary 3.6.

Corollary 3.6. *Let Π' be a CKKS variant whose decryption equation satisfies $\langle \text{ct}, \text{sk} \rangle \bmod q = \mathbf{m} + \mathbf{e} + \mathbf{e}_{\text{aux}}$ with $\|\mathbf{e}_{\text{aux}}\|_\infty \leq B_{\text{aux}}$. Define the augmented noise margins:*

$$\eta'_{\text{sup}} := \alpha((B_{\text{shift}}^{\text{coeff}} + B_{\text{aux}})/\Delta + B_{C2S}^{\text{slot}}) + B_{\text{pmult}}^{\text{slot}} + B_{\text{fresh}}^{\text{slot}},$$

$$\eta'_{\text{sign}} := \eta'_{\text{sup}} + B_{\text{off}}^{\text{slot}}.$$

If Π' satisfies the augmented slot-margin conditions:

$$\eta'_{\text{sign}} < \tau, \quad \alpha - \eta'_{\text{sup}} > \tau, \quad \text{and} \quad 2\alpha - \eta'_{\text{sign}} > \tau,$$

then Algorithm 1 still extracts the secret key s within $2N$ queries.

Note that variants utilizing IND-CPA^D-resistant noise flooding [60], [61] may also conform to this template if the flooding parameters remain below these strict margins; a full analysis under randomized-oracle flooding (where the reaction bit becomes probabilistic) is left for future work.

4. Experimental Validation

We experimentally evaluate the attacks presented in Algorithm 1 and 2 using Lattigo v6.2.0 [56] across various parameter settings.

The implemented attack follows the steps of Algorithm 1 and 2. The oracle decodes the resulting plaintext and returns `valid` if and only if $\|\text{slots}\|_\infty \leq 1$, matching the slot-domain reaction oracle of Section 3.2. The per-query plaintext multiplication does not consume the relinearization key, so the attack needs no `rlk` access.

All experiments were executed using a single-thread on an Intel Core i9-10940X at 3.30 GHz, with 128 GB of RAM, under Ubuntu 20.04.6 LTS.

Parameters. We sweep five parameter triples $(\log N, \log q_0, \log \Delta)$ based on commonly used CKKS parameter regimes: (12, 35, 30), (12, 37, 32), (14, 45, 40), (16, 55, 50), and (16, 60, 58), spanning low- to high-precision settings. They share ring dimensions and level-zero moduli with Table 1, with varying $\log \Delta$ to examine the margin.

For the secret key distribution, we primarily focus on the two standard configurations implemented in the Lattigo library: (i) the uniform ternary distribution $\text{Ternary}(\mathbf{P}: 2/3)$, where each coefficient is non-zero with probability $2/3$ and takes a uniform sign, and (ii) the fixed-weight sparse ternary distribution $\text{Ternary}(\mathbf{H}: h)$, where the number of non-zero coefficients is fixed to a constant $h \ll N$. To demonstrate the comprehensive reach of our attack, we further extend it to additional

distributions, such as Bernoulli-sparse ternary distributions ($\text{Ternary}(\mathbf{P}: \cdot)$) and the discrete Gaussian distribution ($\text{DiscreteGaussian}(\text{Sigma}: \cdot, \text{Bound}: \cdot)$).

For these parameter regimes, we employ Lattigo’s six-prime modulus chains. The $C2S$ layer consumes four levels, leaving an output modulus of at least q_0 , where we evaluate the modulus condition Equation (3). The mask multiplication applies plaintext multiplication without rescaling, so decoding by the ciphertext’s own scale preserves the slot values. The maximum modulus bit is then $\log Q \approx \log q_0 + 50 \times 5 \approx 285\text{--}305$, and $\log PQ \lesssim 427$ for most parameter sets, except for one parameter set at $\log N = 16$ with $\log PQ \approx 472$. The full configurations are provided in the artifact.

Although our attack is independent of the underlying RLWE hardness or the IND-CPA security of the scheme, the evaluated parameters for $\log N \in \{14, 16\}$ with the uniform ternary secrets conform to the standard 128-bit security level per the HE security guidelines [11].

Results of Algorithm 1. We summarize the attack results targeting CKKS with uniform ternary secrets, in Table 2. Every trial recovered the secret in full: across all 160 runs, in a total of $> 3,000,000$ oracle queries, every coefficient was recovered exactly, with no false positives or false negatives from the reaction oracle. The two $\log N = 16$ regimes were each evaluated on five trials owing to their substantially higher runtime; yet together, they comprise over 1,000,000 oracle queries with no error. By construction, the query count is $N + |\mathcal{S}| \approx 1.667N$. This is within a factor 1.05 of the average-case lower bound $H_{2/3}(s) \approx 1.585N$ from Theorem 3.2. End-to-end, the attack completed in 30.0 seconds at $\log N = 12$, 8.4 minutes at $\log N = 14$, and 142 minutes at (16, 55, 50) on a single thread, with a per-trial standard deviation below 2% of the mean. The dominant per-query cost was the plaintext multiplication and decryption.

Robustness across secret distributions. To verify that the attack is not tied to the default uniform ternary secret, we sweep additional secret distributions at (12, 35, 30) and (14, 45, 40): ternary with fixed-weights of $h \in \{128, 192, 256\}$, Bernoulli-sparse ternary with $p \in \{1/3, 1/2, 2/3, 0.9\}$, and a discrete Gaussian with $\sigma = 3.2$ truncated to the integer support $[-S, S]$, where $S = \lceil 6\sigma \rceil = 20$. The results are summarized in Table 3: All 515 runs recovered the full secret. Each of the 490 ternary-secret runs used $N + |\mathcal{S}| < 2N$ queries; the 25 discrete-Gaussian runs used the bounded-secret bisection described in Theorem 3.3.

Against the average-case lower bound $H_p(s)$ of Theorem 3.2, the Bernoulli-sparse reported a factor of ≤ 1.07 for $p \leq 2/3$ and is essentially tight at $p = 1/2$. The discrete Gaussian secret is recovered with $\approx 5.4N$ queries, close to the support-size benchmark $N \log_2(2S + 1) \approx 5.36N$ for the bounded-secret bisection of Remark 3.3. The fixed-weight cases show worse ratios against their exact entropy $H(s) = \log_2 \binom{N}{h} + h$.

Empirical noise. To empirically validate the noise behavior analyzed in our framework, we measured the exact noise

$\log N$	$\log q_0$	$\log \Delta$	trials	$ S $ (mean $\pm$ std, $ S /N$)	$\#Q$ (mean $\pm$ std)	$\#Q/H_{2/3}(s)$	wall (sec, mean $\pm$ std)
12	35	30	50	2,736 $\pm$ 26 (0.668)	6,832 $\pm$ 26	1.052	30.02 $\pm$ 0.14
12	37	32	50	2,730 $\pm$ 34 (0.667)	6,826 $\pm$ 34	1.051	30.06 $\pm$ 0.19
14	45	40	50	10,929 $\pm$ 63 (0.667)	27,313 $\pm$ 63	1.052	506.1 $\pm$ 5.1
16	55	50	5	43,696 $\pm$ 141 (0.667)	109,232 $\pm$ 141	1.052	8,529 $\pm$ 168
16	60	58	5	43,683 $\pm$ 124 (0.667)	109,219 $\pm$ 124	1.051	68,408 $\pm$ 76*

TABLE 2: End-to-end results of Algorithm 1, against Lattigo v6.2.0 (with uniform ternary secrets). Secret keys were fully recovered in every trial. The first four parameters are standard CKKS regimes sharing (N, q_0) with [56]; the last one is in a high-precision setting [11]. The wall time in the last row (*), was obtained from five concurrent trials; hence, is not directly comparable to other single-trial rows. $\#Q$ denotes the query-counts, whose theoretical lower bound is $H_{2/3}(s)$.

Secret distribution	$\log N = 12$		$\log N = 14$	
	trials	$\#Q/H(s)$	trials	$\#Q/H(s)$
ternary, $h = 128$	50	4.470	20	13.724
ternary, $h = 192$	50	3.286	20	9.785
ternary, $h = 256$	50	2.666	20	7.728
ternary, $p = 1/3$	50	1.064	20	1.065
ternary, $p = 1/2$	50	1.000	20	0.999
ternary, $p = 2/3$	50	1.051	20	1.051
ternary, $p = 0.9$	50	1.388	20	1.388
Gaussian, $\sigma = 3.2$	20	1.008*	5	1.009*

TABLE 3: Secret-distribution Robustness Tests at $(12, 35, 30)$ and $(14, 45, 40)$. The sweep covers fixed-weight ternary, Bernoulli-sparse ternary, and discrete-Gaussian secrets; every trial fully recovered the secret. $\#Q$ denotes the query count. For fixed-weight secrets, $H(s) = \log_2 \binom{N}{h} + h$ is the exact entropy; for Bernoulli-sparse secrets, $H(s) = H_p(s)$. For the truncated discrete Gaussian (*), the denominator is the support-size benchmark $N \log_2(2S + 1)$ with $S = 20$, rather than its Shannon entropy.

profiles during the C2S layers, confirming that both the slot-margin and modulus conditions hold with sufficient margins. In the coefficient domain, let $\mathbf{m}^{(2)}$ denote the plaintext carried by the Step-2 ciphertext. The empirical error $\|\mathbf{m}^{(2)} - \Delta \mathbf{s}\|_\infty$ remains strictly below $2^{17.3}$ across the entire parameter sweep, safely falling under the conservative analytic bound of $B_{C2S}^{\text{coeff}} \leq 2^{18}$ employed for the modulus condition in Equation (3).

In the slot domain, the measured noise B_{C2S}^{slot} tops out at $\leq 2^{-12.7}$ at $(12, 35, 30)$ and drops further to, for e.g., $\leq 2^{-32.0}$ at $(16, 55, 50)$. Each reported value represents the worst-case ℓ_∞ maximum captured across all queried coordinates and experimental trials. Under a constant amplitude of $\alpha = 4$, the slot-margin scaling term satisfies $\alpha B_{C2S}^{\text{slot}} \leq 2^{-10.7} \ll \tau = 1$, even under the lowest precision configuration, and decreases by several additional bits when $\log \Delta \geq 30$. This empirically validates Equation (2).

Results of Algorithm 2. Table 4 summarizes the attack

$\log N$	h	$\#Q$	$\#Q/H(s)$	LLL Time (sec.)
12	32	338	1.134	0.4
12	64	598	1.117	3.3
12	128	1,079	1.142	1,521
14	128	1,469	1.221	1,521*
16	128	1,801	1.234	1,521*

TABLE 4: End-to-End Results of Algorithm 2 (with ternary secrets of fixed-weight h). Every run recovered the exact support; then LLL fully recovered the secret keys. Single-core LLL time (in seconds, $q = \min q_0$) was measured only at $\log N = 12$, since it only depends on h (*). $\#Q$ denotes the query count, and $H(s) = \log_2 \binom{N}{h} + h$ is the exact entropy of a uniformly sampled fixed-weight ternary secret.

results based on Algorithm 2. Across $h \in \{32, 64, 128\}$ and $\log N \in \{12, 14, 16\}$, it recovers the exact support, with no false positives or negatives, in $O(h \log(N/h))$ reaction queries: e.g., at $(\log N, h) = (16, 128)$, it requires only 1,801 queries, achieving $36\times$ reduction over the $N + h = 65,664$ queries necessitated by Algorithm 1.

The recovered support restricts the LWE instance to a genuine h -dimensional LWE problem that can be solved directly from the public key material. Specifically, applying the Lenstra-Lenstra-Lovász (LLL) algorithm [33], [59] on a lattice of dimension $2h + 1$ with N samples efficiently and exactly recovered every secret sign for $h \in \{32, 64, 128\}$. The single-core solving time scales steeply with the Hamming weight, taking 0.4 seconds for $h = 32$, 3.3 seconds for $h = 64$, and 25 minutes for $h = 128$, when $\log q = \min \log q_0 \approx 35$.

According to the lattice estimator [2], under the default ternary-secret model, the estimated bit-cost for solving the reduced lattice instances grows from 33.1 bits at $h = 32$ to 43.2 bits at $h = 1,024$, for $\log q = 285$ (the minimum pk modulus Q). Since we fix the modulus to small ≈ 285 bits, these are rough upper bounds: for larger moduli of 700–1,600 bits and for standard Hamming weights of $h \in \{128, 192, 256\}$ [10], [11], the cost further decreases, far below the standard 128-bit level.

5. Countermeasures

Reaction leakage can be addressed at several layers. Below, we present two direct and deployment-compatible measures: standardizing the externally visible response and enforcing per-key reaction monitoring.

Removing the reaction signal. The oracle exists only because the client’s externally visible behavior depends on the validity outcome, so the most robust and deployment-compatible mitigation is to remove that dependence: return the same protocol response regardless of the admissibility check, handling any failure internally. This eliminates the one-bit oracle while changing only outward behavior, not the FHE computation.

Bounding and monitoring reactions. When the response cannot be fully standardized, bounding the number of reactions per secret key is a cheap complementary defense. Key recovery requires resolving enough coordinates that the projected LWE instance falls into lattice-reduction range (e.g., LLL); a sufficiently small cap will leave enough room for security. Even without a hard cap, the attack is detectable: each adversarial trial forces a rejection with high probability, so an elevated per-key rejection rate and query volume provide a natural aggregate detection vector.

Other than the two major directions, we consider three further mitigation strategies. Below, we discuss why each method—despite seeming straightforward to apply—presents its own limitations, rendering them either ineffective as a standalone defense or detrimental to correctness, performance, or efficiency.

Noise and Parameter Modifications. Violating the slot-margin condition (Equation (2)) via parameter tuning is fragile: because this margin is tied to CKKS correctness, increased noise inherently risks rejecting honest evaluations. Protective randomization must therefore target the observable reaction directly, rather than the homomorphic computation pipeline. Crucially, standard IND-CPA^D smudging [44], [60], [61] does not transfer to our setting. Smudging typically adds noise $\|e_{\text{smdg}}\|_{\infty} \ll \Delta$ to mask sub- Δ structures. However, against Algorithm 1, our isolated slot for $s_i \neq 0$ reaches magnitude $\alpha = 4$, quadruple the admissible range $\tau = 1$. Pulling this manipulated slot below τ requires smudging noise of magnitude at least $\alpha - \tau = 3$, which would indiscriminately reject honest outputs. In short, smudging strong enough to thwart our attack is functionally indistinguishable from smudging that destroys correctness, so parameter changes are impractical for deep or iterative workloads.

Changing Rings. Our attack relies strictly on the encryption-key shift $(a - \Delta, b)$ and a public C2S transform, remaining entirely agnostic to the splitting behavior of $x^N + 1$. Migrating from power-of-two cyclotomics to alternative rings [19], [55] is therefore not a standalone defense if an analogous transform exists. Furthermore, this approach severely degrades performance by sacrificing highly optimized NTT pipelines.

Cryptographic Integrity. Upgrading to vCCA-secure FHE [63], [67] or verifiable FHE [4], [9], [36], [37], [38], [40], [43] provides a principled defense, removing the reaction signal by rejecting ciphertexts that are not certified outputs of an intended computation. Recent CKKS-specific verifiable schemes [15], [75] would cleanly reject our crafted ciphertexts, as the attacker cannot prove honest execution of a client-approved circuit. However, these mechanisms introduce a substantial efficiency penalty, making them difficult to integrate into practical deployments.

Overall, the standardization of reaction and per-key reaction monitoring remains the most practical mitigation.

6. Conclusion and Open Questions

We presented a protocol-level reaction attack against CKKS deployments that expose decoded-slot validity responses and C2S rotation keys. Using a simple public-key shift and fresh encryptions of zero, our attack recovers ternary secrets in query complexity that tightly matches theoretical lower bounds. Crucially, by independently randomizing each query, our approach inherently bypasses simple pattern-matching defenses.

Our results demonstrate that vulnerabilities can emerge strictly at the protocol level, without refuting RLWE or IND-CPA security: secure outsourced computation must tightly control how decoded outputs, including single-bit validity reactions, are exposed. The most practical defense is to standardize the client’s externally visible response and monitor reactions per key. The elevated query volume and rejection rate our attack induces further offer a natural detection vector through aggregate failure-rate analysis.

Open Questions. We highlight four directions for future work: (i) Detection Models: Formalizing detector models to determine which aggregate heuristics (e.g., query volume, rejection rates) provably prevent key recovery; (ii) Closing Theoretical Gaps: Reducing the minimal query gap between our theoretical bound and the experimental average for sparse secrets (close to the ≈ 1.05 gap in uniform ternary); (iii) Scheme Generality: Investigating whether similar reaction attacks threaten other widely deployed RLWE schemes, such as BGV and BFV; and (iv) Safe Validation: Designing output-validation mechanisms that support legitimate retries without leaking externally exploitable accept/reject bits.

Acknowledgments

Jiseung Kim is supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2026-25477662), and the 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). Changmin Lee is supported by the Korea Internet & Security Agency

(KISA) grant funded by the Korean government (PIPC) (No. 2780000036, Masters and Doctoral Program for Advanced Talent Development in Personal Data Protection and Responsible Use) and by Korea University grant K2603241. Song Bian is supported by the National Natural Science Foundation of China (62572020).

References

- [1] A. Al Badawi, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee *et al.*, “OpenFHE: Open-source fully homomorphic encryption library,” in *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, 2022, pp. 53–63.
- [2] M. R. Albrecht, R. Player, and S. Scott, “On the concrete hardness of Learning with Errors,” *Journal of Mathematical Cryptology*, vol. 9, no. 3, pp. 169–203, 2015.
- [3] W. Ao and V. N. Boddeti, “AutoFHE: Automated adaption of CNNs for efficient evaluation over FHE,” in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 2173–2190.
- [4] S. Atapoor, K. Bagheri, H. V. L. Pereira, and J. Spiessens, “Verifiable FHE via lattice-based SNARKs,” *IACR Communications in Cryptology*, 2024.
- [5] F. Aydin, E. Karabulut, S. Potluri, E. Alkim, and A. Aysu, “RevEAL: Single-trace side-channel leakage of the SEAL homomorphic encryption library,” in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 1527–1532.
- [6] Y. Bae, J. H. Cheon, G. Hanrot *et al.*, “Fast homomorphic linear algebra with blas,” *Journal of Cryptology*, vol. 39, no. 25, 2026.
- [7] F. Bergamaschi, A. Costache, D. Dachman-Soled, H. Kippen, L. LaBuff, and R. Tang, “Revisiting the security of approximate FHE with noise-flooding countermeasures,” in *Public-Key Cryptography – PKC 2025*. Springer, 2025, pp. 102–134.
- [8] D. Bleichenbacher, “Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS# 1,” in *Advances in Cryptology—CRYPTO’98: 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings 18*. Springer, 1998, pp. 1–12.
- [9] A. Bois, I. Cascudo, D. Fiore, and D. Kim, “Flexible and efficient verifiable computation on encrypted data,” in *Public-Key Cryptography – PKC 2021*. Springer, 2021, pp. 528–558.
- [10] J.-P. Bossuat, J. Troncoso-Pastoriza, and J.-P. Hubaux, “Bootstrapping for approximate homomorphic encryption with negligible failure-probability by using sparse-secret encapsulation,” in *ACNS*, 2022.
- [11] J.-P. Bossuat, R. Cammarota, I. Chillotti, B. R. Curtis, W. Dai, H. Gong, E. Hales, D. Kim, B. Kumara *et al.*, “Security guidelines for implementing homomorphic encryption,” *IACR Communications in Cryptology*, 2025.
- [12] J.-P. Bossuat, C. Mouchet, J. Troncoso-Pastoriza, and J.-P. Hubaux, “Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys,” in *Advances in Cryptology—EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2021, pp. 587–617.
- [13] Z. Brakerski, “Fully homomorphic encryption without modulus switching from classical GapSVP,” in *Advances in Cryptology – CRYPTO 2012*. Springer, 2012, pp. 868–886.
- [14] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(leveled) fully homomorphic encryption without bootstrapping,” *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 1–36, 2014.
- [15] I. Cascudo, A. Costache, D. Cozzo, D. Fiore, A. Guimarães, and E. Soria-Vázquez, “Verifiable computation for approximate homomorphic encryption schemes,” in *Advances in Cryptology – CRYPTO 2025*. Springer, 2025, pp. 643–677.
- [16] B. Chaturvedi, A. Chakraborty, A. Chatterjee, and D. Mukhopadhyay, “A practical full key recovery attack on TFHE and FHEW by inducing decryption errors,” *Cryptology ePrint Archive*, 2022.
- [17] —, ““Ask and Thou Shall Receive”: Reaction-based full key recovery attacks on FHE,” in *Computer Security – ESORICS 2024*. Springer, 2024, pp. 457–477.
- [18] M. Checri, R. Sirdey, A. Boudguiga, and J.-P. Bultel, “On the practical CPA^d security of “exact” and threshold FHE schemes and libraries,” in *Advances in Cryptology – CRYPTO 2024*. Springer, 2024, pp. 3–33.
- [19] H. Chen, K. Lauter, and K. E. Stange, “Attacks on the search RLWE problem with small errors,” *SIAM Journal on Applied Algebra and Geometry*, vol. 1, no. 1, pp. 665–682, 2017.
- [20] M. Chenal and Q. Tang, “On key recovery attacks against existing somewhat homomorphic encryption schemes,” in *Progress in Cryptology – LATINCRYPT 2014*, ser. Lecture Notes in Computer Science, vol. 8895. Springer, 2015, pp. 239–258.
- [21] J. H. Cheon, H. Choe, A. Passetlègue, D. Stehlé, and E. 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*, 2024, pp. 2505–2519.
- [22] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, “Bootstrapping for approximate homomorphic encryption,” in *Advances in Cryptology—EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29–May 3, 2018 Proceedings, Part I 37*. Springer, 2018, pp. 360–384.
- [23] —, “A full RNS variant of approximate homomorphic encryption,” in *Selected Areas in Cryptography—SAC 2018: 25th International Conference, Calgary, AB, Canada, August 15–17, 2018, Revised Selected Papers 25*. Springer, 2019, pp. 347–368.
- [24] J. H. Cheon, G. Hanrot, J. Kim, and D. Stehlé, “SHIP: A shallow and highly parallelizable CKKS bootstrapping algorithm,” in *Advances in Cryptology – EUROCRYPT 2025*. Springer, 2025, pp. 398–428.
- [25] J. H. Cheon, M. Kang, and J. H. Park, “Towards lightweight ckks: On client cost efficiency,” in *Proceedings of the ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS ’26. New York, NY, USA: Association for Computing Machinery, 2026, pp. 1387–1398. [Online]. Available: <https://doi.org/10.1145/3779208.3785268>
- [26] J. H. Cheon, A. Kim, M. Kim, and Y. Song, “Homomorphic encryption for arithmetic of approximate numbers,” in *Advances in Cryptology—ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3–7, 2017, Proceedings, Part I 23*. Springer, 2017, pp. 409–437.
- [27] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, “TFHE: Fast fully homomorphic encryption over the torus,” *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, 2020.
- [28] I. Chillotti, N. Gama, and L. Goubin, “Attacking FHE-based applications by software fault injections,” *Cryptology ePrint Archive*, 2016.
- [29] E. J. Chou, A. Gururajan, K. Laine, N. K. Goel, A. Bertiger, and J. W. Stokes, “Privacy-preserving phishing web page classification via fully homomorphic encryption,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 2792–2796.
- [30] CryptoLab, “HEaaN Library (2022),” <https://heaan.it/>, 2022, accessed: 2025-04-11.
- [31] A. Das, S. Dutta, and A. Adhikari, “Indistinguishability against chosen ciphertext verification attack revisited: The complete picture,” in *Provable Security: 7th International Conference, ProvSec 2013, Melaka, Malaysia, October 23–25, 2013. Proceedings 7*. Springer, 2013, pp. 104–120.
- [32] D.-Z. Du and F. K. Hwang, *Combinatorial Group Testing and Its Applications*, 2nd ed. World Scientific, 2000.

- [33] L. Ducas, L. N. Pulles, and M. Stevens, "Towards a modern LLL implementation," in *ASIACRYPT*, 2025.
- [34] L. Ducas and D. Micciancio, "FHEW: Bootstrapping homomorphic encryption in less than a second," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2015, pp. 617–640.
- [35] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *Cryptology ePrint Archive*, 2012.
- [36] D. Fiore, R. Gennaro, and V. Pastro, "Efficiently verifiable computation on encrypted data," in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, 2014, pp. 844–855.
- [37] D. Fiore, A. Nitulescu, and D. 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. Springer, 2020, pp. 124–154.
- [38] C. Ganesh, A. Nitulescu, and E. Soria-Vazquez, "Rinocchio: SNARKs for ring arithmetic," *Journal of Cryptology*, vol. 36, no. 4, p. 41, 2023.
- [39] R. Geelen and F. Vercauteren, "Fully homomorphic encryption for cyclotomic prime moduli," in *Advances in Cryptology – EUROCRYPT 2025*. Springer, 2025, pp. 366–397.
- [40] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30*. Springer, 2010, pp. 465–482.
- [41] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 169–178.
- [42] B. Ghaleb and W. J. Buchanan, "Side channel analysis in homomorphic encryption," *Cryptology ePrint Archive*, Paper 2025/867, 2025. [Online]. Available: <https://eprint.iacr.org/2025/867>
- [43] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich, "How to run Turing machines on encrypted data," in *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*. Springer, 2013, pp. 536–553.
- [44] Q. Guo, D. Nabokov, E. Suvanto, and T. Johansson, "Key recovery attacks on approximate homomorphic encryption with non-worst-case noise flooding countermeasures," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 7447–7461.
- [45] S. Halevi and V. Shoup, "Faster homomorphic linear transformations in HELib," in *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference*. Springer, 2018, pp. 93–120.
- [46] C. Hall, I. Goldberg, and B. Schneier, "Reaction attacks against several public-key cryptosystems," in *Information and Communication Security: Second International Conference, ICICS'99, Sydney, Australia, November 9-11, 1999. Proceedings 2*. Springer, 1999, pp. 2–12.
- [47] "HElib," <https://github.com/homenc/HElib>, accessed: 2023-10-06.
- [48] Z. Hu, F. Sun, and J. Jiang, "Ciphertext verification security of symmetric encryption schemes," *Science in China Series F: Information Sciences*, vol. 52, no. 9, pp. 1617–1631, 2009.
- [49] F. K. Hwang, "A method for detecting all defective members in a population by group testing," *Journal of the American Statistical Association*, vol. 67, no. 339, pp. 605–608, 1972.
- [50] I. Hwang, S. Min, J. Seo, and Y. Song, "On the security and privacy of CKKS-based homomorphic evaluation protocols," in *Advances in Cryptology – ASIACRYPT 2025*. Springer, 2025, pp. 296–330.
- [51] A. Kim, M. Deryabin, J. Eom, R. Choi, Y. Lee, W. Ghang, and D. Yoo, "General bootstrapping approach for RLWE-based homomorphic encryption," *IEEE Transactions on Computers*, vol. 73, no. 1, pp. 86–96, 2024.
- [52] M. Kim, A. O. Harmanci, J.-P. Bossuat, S. Carpov, J. H. Cheon, I. Chillotti, W. Cho, D. Froelicher, N. Gama, M. Georgieva *et al.*, "Ultrafast homomorphic encryption models enable secure outsourcing of genotype imputation," *Cell systems*, vol. 12, no. 11, pp. 1108–1120, 2021.
- [53] M. Kim, X. Jiang, K. Lauter, E. Ismayilzada, and S. Shams, "Secure human action recognition by encrypted neural network inference," *Nature communications*, vol. 13, no. 1, p. 4799, 2022.
- [54] R. Kumazaki and Y. Kaji, "Reaction attack on TFHE: Minimum number of oracle queries and nearly optimum attacking scheme," in *Computer Security – ESORICS 2025*. Springer, 2025, pp. 247–261.
- [55] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Designs, Codes and Cryptography*, vol. 75, no. 3, pp. 565–599, 2015.
- [56] "Lattigo v6," Online: <https://github.com/tuneinsight/lattigo>, Aug. 2024, ePFL-LDS, Tune Insight SA.
- [57] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim *et al.*, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," *IEEE Access*, vol. 10, pp. 30 039–30 054, 2022.
- [58] J.-W. Lee, E. Lee, Y.-S. Kim, and J.-S. No, "Rotation key reduction for client-server systems of deep neural network on fully homomorphic encryption," in *Advances in Cryptology – ASIACRYPT 2023*, ser. Lecture Notes in Computer Science. Springer, 2023, pp. 36–68.
- [59] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," 1982.
- [60] B. Li and D. Micciancio, "On the security of homomorphic encryption on approximate numbers," in *Advances in Cryptology – EUROCRYPT 2021*. Springer, 2021, pp. 648–677.
- [61] B. Li, D. Micciancio, M. Schultz, and J. Sorrell, "Securing approximate homomorphic encryption using differential privacy," in *Advances in Cryptology – CRYPTO 2022*. Springer, 2022, pp. 560–589.
- [62] Z. Liu and Y. Wang, "Relaxed functional bootstrapping: A new perspective on BGV/BFV bootstrapping," in *Advances in Cryptology – ASIACRYPT 2024*. Springer, 2024, pp. 208–240.
- [63] J. Loftus, A. May, N. P. Smart, and F. Vercauteren, "On CCA-secure somewhat homomorphic encryption," in *Selected Areas in Cryptography: 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers 18*. Springer, 2012, pp. 55–72.
- [64] Q. Lou and L. Jiang, "HEMET: A homomorphic-encryption-friendly privacy-preserving mobile neural network architecture," in *Proceedings of the 38th International Conference on Machine Learning*, 2021, pp. 7102–7110.
- [65] S. Ma, T. Huang, A. Wang, and X. Wang, "Faster BGV bootstrapping for power-of-two cyclotomics through homomorphic NTT," in *Advances in Cryptology – ASIACRYPT 2024*. Springer, 2024, pp. 143–175.
- [66] L. L. Mankali, M. Nabeel, F. Raees, M. Maniatakos, O. Sinanoglu, and J. Knechtel, "GlitchFHE: Attacking fully homomorphic encryption using fault injection," in *34th USENIX Security Symposium (USENIX Security 25)*, 2025, pp. 8481–8500.
- [67] M. Manulis and J. Nguyen, "Fully homomorphic encryption beyond IND-CCA1 security: Integrity through verifiability," in *Advances in Cryptology – EUROCRYPT 2024*. Springer, 2024, pp. 63–93.
- [68] OWASP Foundation, "Input validation cheat sheet," OWASP Cheat Sheet Series, 2024, https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html.
- [69] "PALISADE," <https://gitlab.com/palisade/palisade-release>, accessed: 2023-10-06.
- [70] J. H. Park, "Ciphertext-ciphertext matrix multiplication: Fast for large matrices," in *Advances in Cryptology – EUROCRYPT 2025*, ser. LNCS, vol. 15608, 2025.

- [71] S. Park, J. Byun, and J. Lee, “Privacy-preserving fair learning of support vector machine with homomorphic encryption,” in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 3572–3583.
- [72] “Microsoft SEAL (release 3.0),” <http://sealcrypto.org>, Oct. 2018, microsoft Research, Redmond, WA.
- [73] X. Sun, P. Zhang, J. K. Liu, J. Yu, and W. Xie, “Private machine learning classification based on fully homomorphic encryption,” *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 352–364, 2020.
- [74] M. Walter, “On side-channel and CVO attacks against TFHE and FHEW,” Cryptology ePrint Archive, Paper 2022/1722, 2022. [Online]. Available: <https://eprint.iacr.org/2022/1722>
- [75] X. Zhang, R. Wang, Z. Liu, B. Xiang, Y. Deng, B. Fisch, and X. Lu, “Phalanx: An FHE-friendly SNARK for verifiable computation on encrypted data,” in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2025, pp. 4035–4048.
- [76] Z. Zhang, T. Plantard, and W. Susilo, “Reaction attack on outsourced computing with fully homomorphic encryption schemes,” in *Information Security and Cryptology – ICISC 2011*, H. Kim, Ed. Springer, 2012, pp. 419–436.

Appendix A.

Ethics Considerations

This paper presents a key-recovery attack against a class of CKKS deployments that expose a decoded-output validity reaction together with public evaluation material sufficient for a coefficient-to-slot layer. Because the work develops a new attack technique, it carries misuse risk: an adversary could use the technique to compromise confidentiality in CKKS-based outsourcing protocols that expose such reaction signals.

The risk to real users in our study is limited. All experiments were conducted in controlled local environments using locally generated keys, synthetic data, and local Lattigo instances. We did not target, measure, exploit, or interact with third-party or production systems. We did not collect personal data, use human-subject data, or involve human participants.

The vulnerability we identify is a deployment-surface and protocol-level leakage channel, not a memory-safety or implementation bug in a specific library. The relevant stakeholders include maintainers of FHE libraries such as Lattigo, OpenFHE, and HEaaN, as well as organizations deploying CKKS-based outsourcing protocols whose application surface exposes a per-query validity reaction. We followed coordinated vulnerability disclosure (CVD): we notified the relevant library maintainers and shared the artifact privately before the public release of exploit-ready code.

The paper also provides mitigation guidance. Primary operational mitigations include standardizing externally visible protocol responses to remove the reaction signal entirely and enforcing per-key reaction caps or monitoring anomalous rejection rates. We also analyze three alternative strategies—parameter modifications (including smudging noise), changing cyclotomic rings, and cryptographic integrity mechanisms (vCCA or verifiable FHE)—and explain their limitations in terms of correctness, performance, or computational overhead.

We believe the defensive benefits of publication outweigh the misuse risk. The attack was validated without affecting real users or systems, and the paper clarifies a concrete deployment risk together with actionable mitigation for CKKS deployments.