How (not) to Switch FHE Schemes: Framework and Attacks in the IND-CPA^{D} Model

Giacomo Santato[1,2] and Riccardo Zanotto[1,2]

[1]CISPA, Saarbrücken, Germany
[2]Saarland University, Saarbrücken, Germany
santato.giacomo@gmail.com, riccardo.zanotto@cispa.de

Abstract

In this paper, we study the IND-CPA^{D} security of FHE schemes combined via scheme switching. We introduce a formal framework for capturing security in this setting and identify sufficient conditions under which such combined schemes achieve IND-CPA^{D} security.

We then focus on the specific case of scheme switching from CKKS to exact FHE schemes. We show that the PEGASUS construction [32] does not achieve IND-CPA^{D} security, and provide a proof-of-concept implementation of a key-recovery attack against the CKKS-to-FHEW switching mechanism in the OpenFHE library.

1 Introduction

Fully Homomorphic Encryption (FHE) enables arbitrary computation over encrypted data, allowing users to offload sensitive tasks to untrusted environments without compromising privacy. It is a foundational cryptographic tool that supports a wide variety of tasks while ensuring data confidentiality throughout the whole computation process.

FHE schemes can be broadly categorized according to the types of operations they support most efficiently. Arithmetic-oriented schemes such as BFV [22], BGV [10], and CKKS [16] are designed for the homomorphic evaluation of arithmetic circuits. CKKS supports approximate arithmetic over real and complex numbers, and natively encodes vectors via SIMD packing, enabling faster parallel computations. In contrast, bit-oriented schemes such as TFHE [17] and FHEW [21] are tailored for gate-by-gate evaluation of Boolean circuits and feature fast bootstrapping mechanisms. These schemes support look-up table evaluation and functional bootstrapping.

Since no single FHE scheme is optimal for all types of operations, combining multiple schemes within a single computational pipeline is often desirable. To support such hybrid workflows, several frameworks have introduced *scheme-switching* mechanisms that transform ciphertexts from one FHE scheme to another without decryption. We describe the high-level idea of scheme switching in Figure 1.

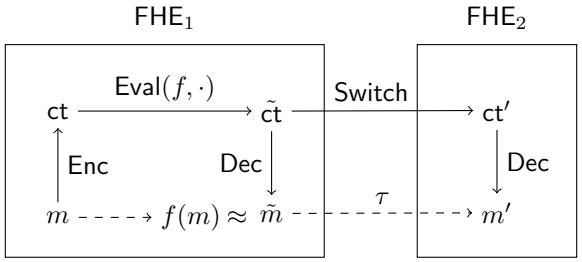

Figure 1: Diagram of the scheme switching functionality

Concrete examples include Chimera [8], which enables switching between CKKS and TFHE/FHEW, as well as between BFV and TFHE; and PEGASUS [32], which supports switching between CKKS and

TFHE/FHEW. These constructions make it possible to offload different parts of a computation to the schemes best suited for each task, thereby improving performance and flexibility.

While these frameworks focus on improving performance and expressiveness, less attention has been paid to how such hybrid designs interact with modern security models. Past works have shown that FHE schemes may leak key-dependent information when decryption outputs are visible to the adversary. Initial attacks focused on CKKS [28], exploiting its approximate decryption mechanism to extract linear relations on the secret key. Subsequent results demonstrated that even exact schemes such as BGV, BFV, and TFHE can be vulnerable in practice, due to decryption failures or residual noise [14, 13]. In all these cases, the adversary observes a decrypted result that differs from the expected plaintext, and this mismatch reveals information that could not have been simulated independently, effectively leaking bits of the underlying secret key. These findings highlight limitations in the classical $\mathsf{IND\text{-}CPA}$ model, which assumes that ciphertexts are public but decrypted outputs are never exposed. This assumption does not reflect many practical FHE settings, where results are explicitly shared, for instance in public statistics or collaborative analytics. To address this gap, Li and Micciancio introduced the $\mathsf{IND\text{-}CPA^D}$ model, which augments $\mathsf{IND\text{-}CPA}$ with a restricted decryption oracle reflecting access to the output of honest homomorphic computations. They showed that CKKS, in its original form, is insecure in this model [28], and later proposed a modified construction that achieves $\mathsf{IND\text{-}CPA^D}$ security [29].

Despite increasing interest in scheme-switching constructions, their security under the $\mathsf{IND\text{-}CPA^D}$ model remains largely unexplored. All known $\mathsf{IND\text{-}CPA^D}$ attacks operate within a single FHE scheme, and no formal analysis exists on how security composes across scheme-switching boundaries.

The PEGASUS paper briefly acknowledges this concern, noting that

> "... the decryptor in PEGASUS should not reveal the decrypted values of CKKS ciphertexts to the encryptor and anyone else without doing any counter-measurement ..." [32, Section II.D.],

but provides no formal model or justification for this claim, and refers only to standard countermeasures [15] on the CKKS decryption.

In this paper we fill this gap in the literature by providing a formal study of the security of FHE pipelines that incorporate *scheme-switching* functionalities.

Our contributions are as follows:

- We construct a framework for studying the security of FHE pipelines. In doing so, we provide formal definitions of the scheme switch algorithm and of an FHE pipeline, and we adapt the existing $\mathsf{IND\text{-}CPA^D}$ definition to this setting.
- We present a key-recovery attack on an FHE pipeline switching from an $\mathsf{IND\text{-}CPA^D}$-secure CKKS scheme to an $\mathsf{IND\text{-}CPA^D}$-secure exact FHE scheme. As a proof of concept, we implement our attack against the CKKS-to-FHEW switching mechanism in the OpenFHE library [3].
- We recognize as a root cause of our attack the lack of *ciphertext hygiene*, that we formally define and is a necessary ingredient in our later results.
- We give a general construction for obtaining $\mathsf{IND\text{-}CPA^D}$-secure pipelines, starting from $\mathsf{IND\text{-}CPA}$-secure ones, via noise flooding and enforcing ciphertext hygiene.
- We show that $\mathsf{IND\text{-}CPA^D}$ of the individual schemes is sufficient to achieve $\mathsf{IND\text{-}CPA^D}$ security for the full pipeline, in settings where the scheme-switching key is private. We show that the same result does not hold in general for when the scheme-switching key is public.
- We formalize the folklore scheme-switching construction and analyze it within our framework.

1.1 Technical Overview

The first main contribution of this paper is the introduction of a formal framework for analyzing FHE scheme switching between two different FHE schemes, FHE_1 and FHE_2.

Many applications already use this concept in the following way: we have a ciphertext ct from FHE_1, and we want to turn it into a new ciphertext $\mathsf{ct}' \leftarrow \mathsf{Switch}(\mathsf{ct})$ for FHE_2.

The main property required by applications is *correctness*, i.e.

$$\mathsf{FHE}_2.\mathsf{Dec}(\mathsf{Switch}(\mathsf{ct})) \approx \mathsf{FHE}_1.\mathsf{Dec}(\mathsf{ct}).$$

Notice that if the message spaces of FHE_1 and FHE_2 are not the same, this requirement is not even well-defined. We thus introduce the notion of *plaintext mapping*, which provides a way to express "equality" of messages in the two different schemes; all the properties of scheme-switching algorithms will then depend on this map. Formally, we fix integers k_1, k_2 and a function

$$\tau : \mathcal{M}_1^{k_1} \to \mathcal{M}_2^{k_2},$$

where $\mathcal{M}_1$ and $\mathcal{M}_2$ are the plaintext spaces of FHE_1 and FHE_2, respectively.

After fixing a mapping τ, we can define *FHE pipelines*, which formalize the concept of scheme switching. In particular, we introduce the *scheme-switching key* ssk, which allows us to compute the switched ciphertext $\mathsf{Switch}(\mathsf{ssk}, \mathsf{ct})$. We require that ssk can be computed from the keys $\mathsf{sk}_i, \mathsf{pk}_i$ of the two schemes via a $\mathsf{SwSetup}$ algorithm.

Notice that our syntax requires that ct is a vector of k_1 ciphertexts of FHE_1, while $\mathsf{Switch}(\mathsf{ssk}, \mathsf{ct})$ is a vector of k_2 ciphertexts of FHE_2.

We can then define the perfect correctness of the switch as

$$\mathsf{FHE}_2.\mathsf{Dec}(\mathsf{sk}_2, \mathsf{Switch}(\mathsf{ssk}, \mathsf{ct})) = \tau(\mathsf{FHE}_1.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct})).$$

Since we mostly deal with approximate FHE schemes, we need a relaxation of this property. In definition 9, we require that, for any message $m \in \mathcal{M}_1^{k_1}$, the approximation error of $\mathsf{Dec}_2(\mathsf{Switch}(\mathsf{ct}))$ with respect to the "plaintext" $\tau(m)$ can be publicly estimated from the approximation error of $\mathsf{Dec}_1(\mathsf{ct})$ with respect to the original message m. This estimate is captured by a function $\mathsf{SwEstimate}$, which we decompose into $\mathsf{SwEst}_{\mathsf{in}}$ and $\mathsf{SwEst}_{\mathsf{out}}$: $\mathsf{SwEst}_{\mathsf{in}}$ captures the uncertainty inherited from the decryption of FHE_1, possibly through its interaction with τ, while $\mathsf{SwEst}_{\mathsf{out}}$ captures the inaccuracy introduced by the decryption of FHE_2.

1.1.1 IND-CPA^{D} Security of FHE Pipelines

We then proceed to study the IND-CPA^{D} security of FHE pipelines, answering the following two natural questions about scheme switching:

How to extend the IND-CPA^{D} *definition to FHE pipelines?*
And how to construct FHE pipelines that achieve this new notion?

In definition 11, we answer the first question. Informally, the pipeline IND-CPA^{D} game gives the adversary the usual encryption, evaluation, and restricted decryption oracles for each scheme, together with an additional switching oracle $\mathcal{SW}$. This oracle moves entries from the source state S_1 to the target state S_2 by applying the plaintext map τ and the scheme switch Switch.

To answer the second question, we first highlight that the IND-CPA^{D} game relies on a correspondence between the plaintext labels and the ciphertexts stored in the states. Without further guarantees on Switch, switching may break this correspondence.

In particular, we give the following generic counterexample, described in detail in Section 4.1. Consider a setting where FHE_1 is an approximate FHE scheme and FHE_2 is exact. Then $\mathcal{SW}$ may insert an entry $(\tau(m_0), \tau(m_1), \mathsf{ct}')$ in S_2 with the property that

$$\mathsf{Dec}_2(\mathsf{ct}') = \tau(\mathsf{Dec}_1(\mathsf{ct})) \notin \{\tau(m_0), \tau(m_1)\}.$$

Such a triple could not arise in the ordinary IND-CPA^{D} game of the exact target scheme. Moreover, it may cause decryption queries to reveal information that is not represented by the stored plaintext labels, thereby allowing an adversary to win the IND-CPA^{D} game of the pipeline.

We call the condition that prevents this issue *ciphertext hygiene*. Informally, ciphertext hygiene requires Switch to return only those switched ciphertext components whose source-side uncertainty is zero. This can be enforced using the public estimate $\rho \leftarrow \mathsf{SwEst}_{\mathsf{in}}(\mathsf{ct})$: if any component ρ_h is greater than zero, then the corresponding ciphertext ct'_h may carry residual uncertainty and must not be returned by Switch; equivalently, it must be discarded by the user.

Now that we have identified ciphertext hygiene as the property needed to keep the pipeline state well formed, we give a generic way to lift IND-CPA-secure pipelines to IND-CPA^{D}-secure ones (cf. Theorem 4). The transformation uses truncated noise flooding, adapted to interact correctly with ciphertext hygiene. The flooding distributions hide the residual decryption error, while the hygiene rule accounts for the additional public uncertainty introduced by flooding. As a result, admissible decryption queries can

be simulated from the plaintext labels alone. Once the real decryption oracles are replaced by these simulators, security reduces to the IND-CPA security of the underlying pipeline.

We also study whether the IND-CPA^{D} security of the individual schemes composes across the switching boundary. We distinguish two cases, depending on whether the scheme-switching key is private or public.

When ssk is private, the adversary can use the switch only through the prescribed switching oracle. In this setting, if the switch behaves as a fresh target encryption of the mapped source plaintext and ciphertext hygiene is enforced, then the IND-CPA^{D} security of the two component schemes lifts to the full pipeline (cf. Theorem 3).

When ssk is public, the situation changes because ssk may contain secret-key-dependent auxiliary information. This additional information is not covered by the stand-alone IND-CPA^{D} guarantees of either component scheme. We provide a counterexample (cf. Section 4.3) showing that security does not, in general, lift to the pipeline in this setting, and that additional properties are needed to control the interaction between the scheme-switching key and the decryption oracles.

1.1.2 Attacks on CKKS-to-Exact Pipelines

In section 5 we describe an attack to the IND-CPA^{D} game of a FHE pipeline where the two schemes are individually IND-CPA^{D}-secure, but the composition is not, due to a poor ciphertext hygiene during the switch operation. In particular, we target the PEGASUS scheme-switching from CKKS to any exact FHE scheme.

The high level idea of the attack is the following: if the scheme-switching algorithm gives us information about *the entire* CKKS plaintext, we can easily adapt the Li-Micciancio attack to recover the CKKS secret key. We exploit the fact that the PEGASUS scheme switch can be modeled by

$$\mathsf{Dec}'(\mathsf{sk}', \mathsf{Switch}(\mathsf{ssk}, \mathsf{ct})) = \tau\left(\mathsf{Decode}_{1/\Delta}(\mathsf{RDec}(\mathsf{sk}, \mathsf{ct}))\right),$$

where on the right we have the *vanilla CKKS decryption*, which leaks part of the underlying RLWE error.

In the actual attack, we start from an encryption $c = (e - a \cdot \mathsf{sk}, a)$ of 0 in CKKS, we multiply it by 2^k for some appropriate k, and then perform scheme switching. We thus obtain ciphertexts encoding the RLWE error e, and by decrypting them in the exact FHE scheme we can recover the error and then the secret key sk.

Notice that a Switch algorithm performing good ciphertext hygiene would instead hide from us the new ciphertexts *adaptively* on k, just as the noise flooding operation in the secured variant of CKKS would do.

1.2 Related Works

1.2.1 Other results on IND-CPA^{D} Security

The IND-CPA^{D} model was introduced by Li and Micciancio, who used it to explain key-recovery attacks on CKKS [28]; Li *et al.* later proposed noise flooding as a countermeasure [29]. Subsequent works showed that IND-CPA^{D} attacks are not limited to approximate schemes, by giving practical attacks and defenses against exact FHE schemes and their threshold variants [14, 13, 27].

Several recent works refine this picture. Guo *et al.* showed that non-worst-case noise flooding can still leave room for attacks, emphasizing the need for robust noise estimates in countermeasures [25]. Bossuat *et al.* and Alexandru *et al.* pursue more practical countermeasures, respectively by studying concrete q-IND-CPA^{D} security and application-aware parameter selection [7, 2].

Other works propose strengthened or alternative formulations of IND-CPA^{D}-style security. Bernard *et al.* introduce ciphertext drift and define a stronger notion, sIND-CPA^{D}, which accounts for adversarially chosen encryption randomness and is strictly stronger than IND-CPA^{D} [4]. Liu, Wang, and Fisch study relaxed functional bootstrapping and introduce IND-CPA-DR, a variant of IND-CPA^{D} in which randomness used during the computation is also exposed to the adversary [31]. Cao, Chatel, and Knabenhans study multi-group homomorphic encryption for on-the-fly MPC and introduce security notions involving partial decryption, circuit privacy, and UC-style composition for imperfect or approximate homomorphic computation [11].

Finally, a broader line of work shows that linearity in decryption, as in vanilla CKKS, is a recurring source of security loss [9, 12].

1.2.2 FHE Scheme Switching

To support hybrid FHE workflows, several frameworks have introduced *scheme-switching* mechanisms that transform ciphertexts from one scheme to another without decryption. Notable examples include Chimera [8], which enables switching between CKKS and TFHE/FHEW, as well as between BFV and TFHE; and PEGASUS [32], which supports switching between CKKS and TFHE/FHEW. Although PEGASUS explicitly warns against revealing CKKS decryption results without countermeasures, it does not provide a formal security analysis under the $\mathsf{IND\text{-}CPA^D}$ model or examine how security composes across switching boundaries.

More recent frameworks extend this line of work. Chameleon [37] focuses on improving the performance of scheme switching through GPU acceleration, while Griffin [34] introduces techniques for handling switching in multi-key settings. However, like their predecessors, these frameworks do not address the compositional security of scheme-switching pipelines in the $\mathsf{IND\text{-}CPA^D}$ setting.

1.2.3 FHE Transciphering

A growing body of work designs block ciphers and other symmetric primitives to interact efficiently with fully homomorphic encryption. These ciphers are either fast to evaluate homomorphically or provide a public procedure that converts a ciphertext of the symmetric scheme into an FHE ciphertext and back; this conversion is often called *FHE transciphering*. Notable examples include HERA [18] and Rubato [26], which are specifically designed to work well with CKKS. See [33] for a survey of constructions that rely on, or enable, FHE transciphering.

Although cryptanalysis of these ciphers is active [30, 24], most studies concentrate on the security of the symmetric primitive itself (covering attacks such as differential analysis, Gröbner basis techniques, lattice methods, etc.) rather than on its interaction with the related FHE scheme. In particular, they rarely address security in the $\mathsf{IND\text{-}CPA^D}$ setting.

More complex protocols are beginning to adopt FHE scheme switching and transciphering as building blocks without analysing $\mathsf{IND\text{-}CPA^D}$ adversaries. The only work we found that explicitly accounts for this model is [19], where the authors design a proxy re-encryption construction and secure it against $\mathsf{IND\text{-}CPA^D}$ attacks.

1.2.4 FHE Compilers

The growing complexity of FHE applications has led to the emergence of *compilers* frameworks, which compile high-level code, typically written in C, C++, or domain-specific languages, into optimized FHE circuits. These systems abstract cryptographic details and generate circuits tailored to different classes of FHE schemes, supporting both arithmetic and boolean operations. Systems such as HEIR [5], Google's FHE Transpiler [23], Porcupine [20], HECO [36], and others aim to improve performance and usability by automating circuit generation.

Some of these compilers, such as HEIR, already plan to support scheme switching between arithmetic-friendly and boolean-friendly schemes, a direction likely to be adopted by other frameworks as well to improve efficiency. As compilers increasingly take responsibility for scheme selection and switching, it becomes essential to understand how $\mathsf{IND\text{-}CPA^D}$ security composes across such transitions, particularly in scenarios where decrypted outputs may be revealed.

2 Preliminaries

Notations. We denote the security parameter by $\lambda \in \mathbb{N}$. A function $\mathsf{negl}(\cdot)$ is negligible if, for every polynomial $p(\cdot)$, there exists k_0 such that for all $k > k_0$, it holds that $\mathsf{negl}(k) < 1/p(k)$.

2.1 Fully Homomorphic Encryption

We recall the definition of Fully Homomorphic Encryption (FHE) in the public-key setting. Formally, a FHE scheme on the message space $\mathcal{M}$ consists of four algorithms $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$, defined as follows:

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$**:** Given a security parameter λ, outputs a secret key sk and a public key pk.

$\mathsf{Enc}(\mathsf{pk}, m) \to \mathsf{ct}$**:** Given a public key pk and a message $m \in \mathcal{M}$, outputs a ciphertext ct.

$\mathsf{Eval}(\mathsf{pk}, f, \mathsf{ct}) \to \mathsf{ct}'$**:** Given a public key pk, a function $f : \mathcal{M}^\ell \to \mathcal{M}$, and a vector of ciphertexts $\mathsf{ct} := \{\mathsf{ct}_i\}_{i\in[\ell]}$, outputs a ciphertext ct'.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m$**:** Given a secret key sk and a ciphertext ct, outputs a message m.

Exact FHE schemes are expected to achieve the following notion.

Definition 1 (Correctness)**.** An FHE scheme $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ is said to be *correct* (or *exact*) if, for every keypair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, every function f, and every vector of ciphertexts $\mathsf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$, it holds that

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, f, \mathsf{ct})) = f(m),$$

where m denotes the vector of plaintexts $\{m_i\}$ obtained by $m_i \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_i)$.

2.1.1 Approximate Correctness

Not all FHE schemes satisfy this perfect correctness property. Some schemes, most notably CKKS, achieve only a much more relaxed variant known as *approximate correctness*. We require the message space to have a norm $\|\cdot\|\colon \mathcal{M} \to \mathbb{R}_{\geq 0}$ in order to define a notion of closeness of messages. In particular, given a ciphertext ct and an "intended" message m, we can define the *error* of that ciphertext w.r.t. a secret key in the following way.

Definition 2 (Ciphertext Error)**.** Let $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be an FHE scheme with message space $\mathcal{M}$, which is a normed space equipped with a norm $\|\cdot\|\colon \mathcal{M} \to \mathbb{R}_{\geq 0}$. For any ciphertext ct, secret key sk, and message m, we define

$$\mathsf{Error}(\mathsf{ct}, m, \mathsf{sk}) := \|\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) - m\|.$$

We first define the approximate correctness of the FHE scheme as an encryption scheme, i.e. what is the error of freshly encrypted ciphertexts.

Definition 3 (Approximate Encryption Correctness)**.** Let $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be an FHE scheme with normed message space $\mathcal{M}$. Given a constant $B \in \mathbb{R}_{\geq 0}$, the $(\mathsf{Enc}, \mathsf{Dec})$ pair is said to be *B-correct* if, for all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and for all messages $m \in \mathcal{M}$, the following property holds:

$$\mathsf{Error}(\mathsf{Enc}(\mathsf{pk}, m), m, \mathsf{sk}) \leq B.$$

We then turn to define approximate correctness of the Eval function, i.e. how far the decryption of some evaluation falls from the plaintext computation.

Definition 4 (Approximate Evaluation Correctness)**.** Let $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be an FHE scheme with normed message space $\mathcal{M}$. The scheme FHE is said to be *approximately correct* if, for all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, for all functions f, and for all vectors of ciphertexts $\mathsf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and plaintexts $m = (m_1, \ldots, m_k)$ such that $\mathsf{Error}(\mathsf{ct}_i, m_i, \mathsf{sk}) \leq t_i$, the following property holds:

$$\mathsf{Error}(\mathsf{Eval}(\mathsf{pk}, f, \mathsf{ct}), f(m), \mathsf{sk}) \leq \mathsf{Estimate}(f, t),$$

where $\mathsf{Estimate}$ is a fixed, publicly computable function, taking as input the function f and the vector of error upper bounds $t = (t_1, \ldots, t_k)$.

Notice that if $B = 0$ and $\mathsf{Estimate}(f, 0) = 0$ for any function f and ciphertext ct, then we recover the notion of an *exact* FHE scheme.

When speaking of approximate FHE schemes, we follow [29] to incorporate the error bound t as part of its ciphertext ct. In particular we denote by $\mathsf{ct}.t$ the *tag* corresponding to a ciphertext, which is updated accordingly by the Eval procedure, using the $\mathsf{Estimate}$ function.

2.2 IND-CPA^{D} Security

We now recall the formal definition of IND-CPA^{D} security from [28].

Definition 5 (q-IND-CPA^{D} security)**.** Let $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be an approximate fully homomorphic encryption scheme. The q-IND-CPA^{D} security experiment $\mathsf{Exp}_b^{q\text{-IND-CPA}^{\mathsf{D}}}$, defined with respect to an adversary $\mathcal{A}$, provides the adversary with access to three stateful oracles $\mathcal{E}, \mathcal{H}, \mathcal{D}$ described in Algorithm 1.

$$\begin{aligned}\mathsf{Exp}_b^{q\text{-IND-CPA}^{\mathsf{D}}}[\mathcal{A}](1^\lambda): \ & (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda),\\ & b' \leftarrow \mathcal{A}^{\mathcal{E}^b_{\mathsf{pk}}, \mathcal{H}_{\mathsf{pk}}, \mathcal{D}_{\mathsf{sk}}}(1^\lambda, \mathsf{pk}),\\ & \textbf{return } b'.\end{aligned}$$

We say that FHE is q-IND-CPA^{D}-secure if for every PPT adversary $\mathcal{A}$ making at most q queries to oracle $\mathcal{D}$ we have that

$$\mathsf{adv}^{q\text{-IND-CPA}^{\mathsf{D}}}[\mathcal{A}](1^\lambda) = \left|\Pr[\mathsf{Exp}_b^{q\text{-IND-CPA}^{\mathsf{D}}}[\mathcal{A}](1^\lambda) = b] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

Algorithm 1: Oracles for the IND-CPA^{D} game

Oracle $\mathcal{E}^b_{\mathsf{pk}}(m_0, m_1)$**:**
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$
- $S[i] \leftarrow (m_0, m_1, \mathsf{ct})$
- $i \leftarrow i + 1$
- **return** ct

Oracle $\mathcal{D}_{\mathsf{sk}}(i)$**:**
- **if** $S[i].m_0 = S[i].m_1$ **then**
  - **return** $\mathsf{Dec}(\mathsf{sk}, S[i].\mathsf{ct})$
- **else**
  - **return** $\perp$
- **end**

Oracle $\mathcal{H}_{\mathsf{pk}}(g, J = (j_1, \ldots, j_k))$**:**
- $\tilde{\mathsf{ct}} \leftarrow \mathsf{Eval}(\mathsf{pk}, g, (S[j_1].\mathsf{ct}, \ldots, S[j_k].\mathsf{ct}))$
- $gm_0 \leftarrow g(S[j_1].m_0, \ldots, S[j_k].m_0)$
- $gm_1 \leftarrow g(S[j_1].m_1, \ldots, S[j_k].m_1)$
- $S[i] \leftarrow (gm_0, gm_1, \tilde{\mathsf{ct}})$
- $i \leftarrow i + 1$
- **return** $\tilde{\mathsf{ct}}$

2.3 Noise Flooding

We recall the noise flooding mechanism from [29], in a truncated variant.

Definition 6 (Truncated noise-flooding mechanism)**.** Let $\mathcal{M}$ be a normed space. A truncated noise-flooding mechanism over $\mathcal{M}$ is a family of random maps

$$\mathsf{NF}_{t,\lambda} : \mathcal{M} \to \mathcal{M},$$

indexed by public error bounds $t \in \mathbb{R}_{\geq 0}$ and security parameter $\lambda \in \mathbb{N}$, satisfying the following two properties.

1. **Uniform shift hiding.** There exists a negligible function $\epsilon^{\mathsf{NF}}(\lambda)$ such that for every $\lambda \in \mathbb{N}$, $t \in \mathbb{R}_{\geq 0}$, and every $x_0, x_1 \in \mathcal{M}$, we have that
$$\|x_0 - x_1\| \leq t \implies \Delta(\mathsf{NF}_{t,\lambda}(x_0), \mathsf{NF}_{t,\lambda}(x_1)) \leq \epsilon^{\mathsf{NF}}(\lambda).$$
2. **Bounded displacement.** There exists a public function $R : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that, for every t and every $x \in \mathcal{M}$,
$$\Pr[\|\mathsf{NF}_{t,\lambda}(x) - x\| \leq R(t)] = 1.$$
For vectors of tags $t = (t_1, \ldots, t_k)$, we write $R(t) := (R(t_1), \ldots, R(t_k))$.

When it is clear from the context, we simply use NF_t as a notation.

2.4 Circuit Privacy

Definition 7 (Circuit privacy)**.** Let $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be a FHE scheme with plaintext space $\mathcal{M}$. We say that FHE is *circuit private* if, for every supported function $f : \mathcal{M}^\ell \to \mathcal{M}$ and every plaintext vector $\mathbf{m} = (m_1, \ldots, m_\ell) \in \mathcal{M}^\ell$, the following two distributions are indistinguishable:

$$\mathsf{Eval}(\mathsf{pk}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \approx_s \mathsf{Enc}(\mathsf{pk}, f(\mathbf{m})),$$

where

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda), \qquad \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, m_i) \quad \text{for } i \in [\ell].$$

The probability is over key generation, encryption, and the randomness of Eval.

2.5 RLWE Encryption

We denote by $\mathcal{R}$ the polynomial ring $\mathcal{R} = \mathbb{Z}_q[X]/(X^n + 1)$, where q is a large ciphertext modulus and n is a power-of-two ring dimension. We also denote as $\Phi_n(X)$ the n-th cyclotomic polynomial. To simplify notation, we identify an element $r \in \mathcal{R}$ with the vector of its coefficients; thus, $r[j]$ denotes the j-th coefficient of the polynomial r for each $j < n$.

A basic RLWE encryption of a message $m \in \mathcal{R}$ under the secret key $\mathsf{sk} \in \mathcal{R}$ is given by $(b, a) = (-\mathsf{sk} \cdot a + m + e,\ a)$, where $a \in \mathcal{R}$ is chosen uniformly at random, and the error polynomial e is sampled independently with coefficients drawn from an error distribution χ_e.

We denote an RLWE encryption of m under the secret key sk by $\mathsf{REnc}(\mathsf{sk}, m)$. Given a ciphertext $\mathsf{ct} = (b, a)$, its decryption is defined as $\mathsf{RDec}(\mathsf{sk}, \mathsf{ct}) = b + a \cdot \mathsf{sk}$.

2.6 CKKS

The CKKS [16] scheme enables approximate homomorphic computation over vectors of complex numbers. It operates over a cyclotomic polynomial ring $\mathcal{R} = \mathbb{Z}_q[X]/(X^n + 1)$. The elements of this ring can be viewed either as polynomials or, via the canonical embedding, as complex vectors.

2.6.1 Canonical Embedding

Let $\omega = e^{\pi i/n}$ be the primitive $2n$-th root of unity, and let $\{\omega^1, \omega^3, \ldots, \omega^{2n-1}\}$ be the roots of $\Phi_{2n}(X) = X^n + 1$ corresponding to the multiplicative group $\mathbb{Z}_n^*$.

Then, given any polynomial $r \in \mathbb{R}[X]/(X^n + 1)$, the map

$$r \mapsto \left(r(\omega^1), r(\omega^3) \ldots, r(\omega^{2n-1})\right) \in \mathbb{C}^n$$

is well defined; moreover, by Hermitian symmetry, we have that $r(\omega^{2n-i}) = \overline{r(\omega^i)}$, thus the image of this map lives in an $n/2$-dimensional subspace of $\mathbb{C}^n$, namely $\{z \in \mathbb{C}^n \ : \ z_{2n-i} = \bar{z_i}\}$.

We define the *canonical embedding* $\psi : \mathbb{R}[X]/(X^n + 1) \to \mathbb{C}^{n/2}$ via the complex evaluation at half of these roots

$$\psi(r) = \left(r(\omega^1), r(\omega^3), \ldots, r(\omega^{n-1})\right).$$

The inverse map $\psi^{-1}(z)$, for a $z \in \mathbb{C}^{n/2}$, is defined by finding an interpolating polynomial such that $r(\omega^{2i-1}) = z_i$ and $r(\omega^{2n-2i+1}) = \bar{z_i}$ for $i = 1, \ldots, n/2$.

2.6.2 Encoding and Decoding

The CKKS scheme has plaintext space $\mathbb{C}^{n/2}$, but uses RLWE encryption for security, so it needs a way to map complex vectors to polynomials in $\mathcal{R}$ and vice-versa. This is done via the encoding and decoding functions, relying on the canonical embedding ψ, identifying $\mathbb{R}[X]/(X^n + 1)$ and $\mathcal{R}$ via rounding.

To encode a plaintext vector $m \in \mathbb{C}^{n/2}$, CKKS applies the inverse of the canonical embedding ψ^{-1}, scales the result by a factor $\Delta > 1$, and rounds to the closest integer to get an element of $\mathcal{R}$:

$$\mathsf{Encode}_\Delta(m) := \mathrm{Round}(\Delta \cdot \psi^{-1}(m)) \in \mathcal{R}.$$

Decoding reverses this procedure: the polynomial is scaled down and mapped back to the complex domain via ψ,

$$\mathsf{Decode}_{1/\Delta}(r) := \psi(\Delta^{-1} \cdot r) \in \mathbb{C}^{n/2}.$$

Due to rounding and the presence of noise in encrypted settings, this process is only approximately correct.

We define the canonical embedding norm of an element $r \in \mathcal{R}$ as $\|r\|_{\text{can}} := \|\psi(r)\|_2$, where ψ is the canonical embedding. This norm is used to track the ciphertext error of CKKS ciphertexts throughout the paper.

2.6.3 Encryption and Decryption

Once the message $m \in \mathbb{C}^{n/2}$ has been encoded into a plaintext polynomial $\mathsf{pt} = \mathsf{Encode}_\Delta(m) \in \mathcal{R}$, encryption proceeds using an RLWE-based encryption scheme under a secret key $\mathsf{sk} \in \mathcal{R}$. The resulting ciphertext is a pair:

$$\mathsf{ct} = \mathsf{REnc}(\mathsf{sk}, \mathsf{pt}) = (b, a) = (-a \cdot \mathsf{sk} + \mathsf{pt} + e, a),$$

where $a \in \mathcal{R}$ is sampled uniformly at random and $e \in \mathcal{R}$ is drawn from a chosen bounded noise distribution (e.g., a discrete Gaussian).

To decrypt a ciphertext $\mathsf{ct} = (b, a)$, the decryptor computes the RLWE decryption to get:

$$\mathsf{RDec}(\mathsf{sk}, \mathsf{ct}) = b + a \cdot \mathsf{sk} = \mathsf{pt} + e.$$

This returns a noisy approximation of the encoded polynomial pt. To recover an approximation of the original message, the decryptor applies the decoding map:

$$m' = \mathsf{Decode}_{1/\Delta}(\mathsf{pt} + e) = \psi(\Delta^{-1} \cdot (\mathsf{pt} + e)),$$

which outputs a vector $m' \in \mathbb{C}^{n/2}$ that approximates m, up to errors induced by the RLWE encryption noise and rounding procedures.

We refer to this decryption algorithm as the *vanilla* CKKS decryption. Recall that, as shown by Li and Micciancio [28], this version of CKKS is not $\mathsf{IND\text{-}CPA^D}$ secure; we briefly recall the attack and countermeasures in section 2.7.

2.7 Previous $\mathsf{IND\text{-}CPA^D}$ Attacks to CKKS and Countermeasures

The first attack exploiting the $\mathsf{IND\text{-}CPA^D}$ model was introduced by Li and Micciancio [28], alongside the definition itself. They showed that for approximate schemes such as CKKS, the decrypted plaintext reveals not only the intended message but also a residual noise term that depends linearly on the secret key.

In particular, they exploit the fact that the vanilla CKKS decryption is almost linear in the secret key, because the decryption of $\mathsf{ct} = (b, a)$ under the secret key sk is $\psi(\Delta^{-1} \cdot (b + a \cdot \mathsf{sk}))$. By observing this linear noise across $\Theta(n)$ carefully chosen ciphertexts, where n is the dimension of the RLWE ring $\mathcal{R}$, a passive adversary can recover the full secret key using standard linear algebra.

The literature proposes two principal strategies for securing CKKS against $\mathsf{IND\text{-}CPA^D}$ adversaries. These strategies, along with their efficiency trade-offs and practical optimizations, are analyzed in depth in [7, 2], which also extend the discussion to countermeasures for exact FHE schemes targeted by other $\mathsf{IND\text{-}CPA^D}$ attacks.

2.7.1 Exact Decryption

The first approach seeks to prevent any information about the ciphertext error e, and hence about the secret key, from ever reaching the decryptor's output. This is achieved by requiring the evaluator to track the worst-case noise growth of each ciphertext throughout homomorphic computation. Whenever the estimated error approaches the correctness threshold $\Delta/2$, the ciphertext must be bootstrapped to reduce its noise before further evaluation. As a result, the decryption output becomes independent of the error term e, closing the leakage channel exploited in the original Li–Micciancio attack.

In this setting, correctness must be treated as a security requirement. This implies that decryption failure, caused by excessive noise, must occur with negligible probability, typically below 2^{-128}. Moreover, this bound must hold even in an *adversarial* setting, where the ciphertexts may be deliberately crafted to maximize noise. This is a much stricter condition than the standard 2^{-80} threshold assumed for benign, non-adversarial workloads.

The main drawback of this approach is performance: ensuring correctness under adversarial conditions requires more frequent bootstrapping and possibly larger ciphertext moduli, significantly increasing latency and memory consumption. The work of [13] estimates that bootstrapping overhead under these conditions can increase by as much as 50%.

2.7.2 Noise Masking

This alternative defence also targets the leakage of information related to the ciphertext error e, but preserves the approximate nature of CKKS. It does so by injecting an additional masking term during decryption. Instead of returning the raw value $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = \mathsf{pt} + e$, the decryptor outputs $\mathsf{Dec}'(\mathsf{sk}, \mathsf{ct}) = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) + \eta = \mathsf{pt} + e + \eta$, where η is a freshly sampled polynomial in $\mathcal{R}$.

The masking noise η is drawn from an integer Gaussian distribution whose variance depends on the worst-case canonical norm estimate t tracked during homomorphic evaluation. Specifically, the noise is sampled from

$$\mathcal{N}_{\mathbb{Z}^n}\left(0,\, \sigma^2 \cdot t^2\right),$$

where $\sigma = 8\sqrt{qn2^{\lambda/2}}$ and q is the bound on the number of queries allowed to the adversary. This suffices to achieve λ-bit $\mathsf{IND\text{-}CPA^D}$ security as shown in [29, Theorem 3].

Since the noise is added only after the computation is complete, this defence does not affect homomorphic execution time. Its runtime cost is minimal, limited to sampling η and performing a single addition. The primary trade-off is a reduction in the precision of the final result.

To ensure robustness against repeated decryption queries, several deployment variants are possible. The decryptor may sample a fresh η per query, restrict decryption to a single call per ciphertext, or use deterministic rounding-based masking to prevent averaging attacks. All variants aim to ensure that no information about e can be inferred from the decrypted outputs.

In the stand-alone CKKS setting, both noise masking and exact decryption successfully prevent known $\mathsf{IND\text{-}CPA^D}$ attacks. Depending on the application, one may prefer to incur overhead in evaluation time or to tolerate reduced output precision.

3 A Framework for Scheme-Switching FHE

In this section we will formally introduce the syntax and properties required by the scheme switch algorithm, both for correctness and for security. We extend the $\mathsf{IND\text{-}CPA^D}$ game to this setting, and in Section 4 we give composition theorems for achieving this notion.

3.1 Formally Defining Scheme Switching

3.1.1 Plaintext Mapping

Formally, we consider two schemes FHE_1 and FHE_2 with respective message spaces $\mathcal{M}_i$ and ciphertext spaces $\mathcal{C}_i$. We define the notion of *plaintext mapping*, which consists of a function $\tau : \mathcal{M}_1^{k_1} \to \mathcal{M}_2^{k_2}$, where the multiplicities k_1 and k_2 depend on the relative interaction of the two schemes and their relative message spaces.

An example of this is when switching from CKKS to TFHE, when only the top l most significant bits of the real part of the message are kept. In this case $\mathcal{M}_1 = \mathbb{C}^{n/2}$, $\mathcal{M}_2 = \mathbb{Z}_2$, with $k_1 = 1$ and $k_2 = l \cdot n/2$.

3.1.2 Scheme Switch

We now give a formal definition of scheme switch.

Definition 8 (Scheme Switch)**.** A scheme switch from FHE_1 to FHE_2 with plaintext mapping $\tau : \mathcal{M}_1^{k_1} \to \mathcal{M}_2^{k_2}$ consists of two algorithms defined as follows:

- $\mathsf{SwSetup}(\mathsf{sk}_1, \mathsf{pk}_2) \to \mathsf{ssk}$: Given a secret key sk_1 for FHE_1 and a public key pk_2 for FHE_2, outputs a *scheme-switching key* ssk.
- $\mathsf{Switch}(\mathsf{ssk}, \mathsf{ct}) \to \mathsf{ct}'$: Given a scheme-switching key ssk and a vector of k_1 ciphertexts $\mathsf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_{k_1})$ in FHE_1, outputs a vector of k_2 ciphertexts $\mathsf{ct}' = (\mathsf{ct}'_1, \ldots, \mathsf{ct}'_{k_2})$ in FHE_2.

We can transpose the correctness definitions to the scheme switch algorithm.

Definition 9 (Approximate Scheme Switch Correctness)**.** Let Switch be a scheme switch from FHE_1 to FHE_2 with plaintext mapping $\tau : \mathcal{M}_1^{k_1} \to \mathcal{M}_2^{k_2}$. We say that Switch is *approximately correct* if there exist two publicly computable functions

$$\mathsf{SwEst}_{\mathsf{in}} : \mathbb{R}_{\geq 0}^{k_1} \to \mathbb{R}_{\geq 0}^{k_2} \quad \text{and} \quad \mathsf{SwEst}_{\mathsf{out}} : \mathcal{C}_1^{k_1} \to \mathbb{R}_{\geq 0}^{k_2}$$

satisfying the following properties.

For every honestly generated key pairs and ssk, for every vector of ciphertexts $\mathsf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_{k_1}) \in \mathcal{C}_1^{k_1}$, and for every vector of plaintexts $m = (m_1, \ldots, m_{k_1}) \in \mathcal{M}_1^{k_1}$ such that $\mathsf{FHE}_1.\mathsf{Error}(\mathsf{ct}_i, m_i, \mathsf{sk}_1) \leq t_i$ the following two inequalities hold component-wise.

First, the input-side uncertainty is bounded by $\mathsf{SwEst}_{\mathsf{in}}$:

$$\|\tau(\mathsf{FHE}_1.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct}))_h - \tau(m)_h\| \leq \mathsf{SwEst}_{\mathsf{in}}(t)_h.$$

Second, the output-side inaccuracy is bounded by $\mathsf{SwEst}_{\mathsf{out}}$:

$$\|\mathsf{FHE}_2.\mathsf{Dec}(\mathsf{sk}_2, \mathsf{Switch}(\mathsf{ssk}, \mathsf{ct})_h) - \tau(\mathsf{FHE}_1.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct}))_h\| \leq \mathsf{SwEst}_{\mathsf{out}}(\mathsf{ct})_h.$$

We denote by $\mathsf{SwEstimate}$ the pair $(\mathsf{SwEst}_{\mathsf{in}}, \mathsf{SwEst}_{\mathsf{out}})$.

It is immediate to see that combining the inbound and outbound switch estimates we are able to control the distance between $\mathsf{FHE}_2.\mathsf{Dec}(\mathsf{sk}_2, \mathsf{Switch}(\mathsf{ssk}, \mathsf{ct}))$ and $\tau(m)$ for any plausible plaintext m of ct. We give a formal result in Lemma 1.

Lemma 1 (Combined switch estimate)**.** *Let* Switch *be an approximately correct scheme switch in the sense of definition 9. Then*

$$\mathsf{SwEstimate}(\mathsf{ct}, t) := \mathsf{SwEst}_{\mathsf{out}}(\mathsf{ct}) + \mathsf{SwEst}_{\mathsf{in}}(t)$$

is a valid switch estimate in the sense that, for every honestly generated key pairs, every $\mathsf{ssk} \leftarrow \mathsf{SwSetup}(\mathsf{sk}_1, \mathsf{pk}_2)$*, every ciphertext vector* $\mathsf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_{k_1})$*, and every plaintext vector* $m = (m_1, \ldots, m_{k_1})$ *satisfying*

$$\mathsf{FHE}_1.\mathsf{Error}(\mathsf{ct}_i, m_i, \mathsf{sk}_1) \leq t_i \qquad \text{for every } i \in \{1, \ldots, k_1\},$$

we have

$$\mathsf{FHE}_2.\mathsf{Error}(\mathsf{Switch}(\mathsf{ssk}, \mathsf{ct}), \tau(m), \mathsf{sk}_2) \leq \mathsf{SwEstimate}(\mathsf{ct}, t),$$

component-wise.

Proof. Fix a component $h \in \{1, \ldots, k_2\}$. By the triangle inequality,

$$\begin{aligned}
&\|\mathsf{FHE}_2.\mathsf{Dec}(\mathsf{sk}_2, \mathsf{Switch}(\mathsf{ssk}, \mathsf{ct})_h) - \tau(m)_h\| \\
&\leq \|\mathsf{FHE}_2.\mathsf{Dec}(\mathsf{sk}_2, \mathsf{Switch}(\mathsf{ssk}, \mathsf{ct})_h) - \tau(\mathsf{FHE}_1.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct}))_h\| \\
&\quad + \|\tau(\mathsf{FHE}_1.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct}))_h - \tau(m)_h\|.
\end{aligned}$$

The first term is bounded by $\mathsf{SwEst}_{\mathsf{out}}(\mathsf{ct})_h$ by output-side switch inaccuracy. The second term is bounded by $\mathsf{SwEst}_{\mathsf{in}}(t)_h$ by input-side uncertainty. Therefore

$$\|\mathsf{FHE}_2.\mathsf{Dec}(\mathsf{sk}_2, \mathsf{Switch}(\mathsf{ssk}, \mathsf{ct})_h) - \tau(m)_h\| \leq \mathsf{SwEst}_{\mathsf{out}}(\mathsf{ct})_h + \mathsf{SwEst}_{\mathsf{in}}(t)_h.$$

Since this holds for every component h, we obtain

$$\mathsf{FHE}_2.\mathsf{Error}(\mathsf{Switch}(\mathsf{ssk}, \mathsf{ct}), \tau(m), \mathsf{sk}_2) \leq \mathsf{SwEst}_{\mathsf{out}}(\mathsf{ct}) + \mathsf{SwEst}_{\mathsf{in}}(t) = \mathsf{SwEstimate}(\mathsf{ct}, t),$$

component-wise. □

Finally, we give the definition of a FHE pipeline to refer to a pair of FHE schemes and the scheme switch between them.

Definition 10 (FHE pipeline)**.** We define a *FHE pipeline* as a triple $\Pi = (\mathsf{FHE}_1, \mathsf{FHE}_2, \mathsf{Switch})$ where FHE_1 and FHE_2 are approximate FHE schemes and Switch is an approximately correct scheme switch algorithm from FHE_1 to FHE_2, with corresponding $\mathsf{SwSetup}$ and $\mathsf{SwEstimate}$ algorithms.

3.2 Extending IND-CPA^{D} to Scheme Switching

We now formalize security by introducing the notion of IND-CPA^{D} for FHE pipelines.

3.2.1 The $\mathcal{SW}$ oracle

To model homomorphic executions that cross the scheme-switching boundary, we define a new oracle $\mathcal{SW}$, described in Algorithm 2.

Algorithm 2: Scheme Switch Oracle

Oracle $\mathcal{SW}_{\mathsf{ssk}}(J = (j_1, \ldots, j_{k_1})$**:**
- $\mathsf{ct} \leftarrow (S_1[j_1].\mathsf{ct}, \ldots, S_1[j_{k_1}].\mathsf{ct})$;
- $\mathsf{ct}' \leftarrow \mathsf{Switch}(\mathsf{ssk}, \mathsf{ct})$;
- $m'_0 \leftarrow \tau(S_1[j_1].m_0, \ldots, S_1[j_{k_1}].m_0)$;
- $m'_1 \leftarrow \tau(S_1[j_1].m_1, \ldots, S_1[j_{k_1}].m_1)$;
- **for** $h \in \{1, \ldots, k_2\}$ **do**
  - $S_2[i_2] \leftarrow (m'_{0,h}, m'_{1,h}, \mathsf{ct}'_h)$;
  - $i_2 \leftarrow i_2 + 1$;
- **end**
- **return** ct';

Algorithm 3: Experiment for Definition 11.

Experiment $\mathsf{Exp}_b^{(q_1,q_2)\text{-}\mathsf{IND\text{-}CPA}^{\mathsf{D}}}[\mathcal{A}](1^\lambda)$**:**
- $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{FHE}_1.\mathsf{KeyGen}(1^\lambda)$
- $(\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{FHE}_2.\mathsf{KeyGen}(1^\lambda)$
- $\mathsf{ssk} \leftarrow \mathsf{SwSetup}(\mathsf{sk}_1, \mathsf{pk}_2)$
- $b' \leftarrow \mathcal{A}^{\mathcal{E}^b_{1,\mathsf{pk}_1}, \mathcal{H}_{1,\mathsf{pk}_1}, \mathcal{D}_{1,\mathsf{sk}_1}, \mathcal{E}^b_{2,\mathsf{pk}_2}, \mathcal{H}_{2,\mathsf{pk}_2}, \mathcal{D}_{2,\mathsf{sk}_2}, \mathcal{SW}_{\mathsf{ssk}}}(1^\lambda, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ssk})$
- **return** b'

3.2.2 IND-CPA^{D} for FHE Pipelines

The $\mathcal{SW}$ oracle enables a formal definition of IND-CPA^{D} security for FHE pipelines.

Definition 11 (IND-CPA^{D} Security for FHE Pipelines)**.** Let $\Pi = (\mathsf{FHE}_1, \mathsf{FHE}_2, \mathsf{Switch})$ be a FHE pipeline. The (q_1, q_2)-IND-CPA^{D} security experiment $\mathsf{Exp}_b^{(q_1,q_2)\text{-}\mathsf{IND\text{-}CPA}^{\mathsf{D}}}$, defined with respect to an adversary $\mathcal{A}$, is described in Algorithm 3. It provides the adversary with access to the three stateful oracles $\mathcal{E}, \mathcal{H}, \mathcal{D}$ described in Algorithm 1, for each of the two FHE scheme, and to the stateful oracle $\mathcal{SW}$ described in Algorithm 2.

We say that the FHE pipeline $\Pi = (\mathsf{FHE}_1, \mathsf{FHE}_2, \mathsf{Switch})$ is (q_1, q_2)-IND-CPA^{D}-secure if for every PPT adversary $\mathcal{A}$ making at most q_1 and q_2 queries to $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively, the following holds

$$\mathsf{adv}_{\Pi}^{(q_1,q_2)\text{-}\mathsf{IND\text{-}CPA}^{\mathsf{D}}}[\mathcal{A}](1^\lambda) = \left| \Pr[\mathsf{Exp}_b^{(q_1,q_2)\text{-}\mathsf{IND\text{-}CPA}^{\mathsf{D}}}[\mathcal{A}](1^\lambda) = b] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

3.2.3 Weak version without ssk

The definition above corresponds to a strong threat model in which ssk is explicitly given to the adversary. Depending on the deployment setting, one may also consider a weaker variant in which the adversary is not given direct access to ssk.

This weaker model is closer in spirit to the passive-adversary setting of Li and Micciancio's original IND-CPA^{D} formulation, where the adversary is not the server itself but has only passive access to the communication and final output channels, i.e., it can eavesdrop, but is not allowed to tamper with ciphertexts or alter the final result of the computation, therefore does not need to be handed additional material to evaluate.

We denote this weakened version of the IND-CPA^{D} notion for FHE pipelines by w-IND-CPA^{D}, i.e. we give the adversary only $\mathsf{pk}_1, \mathsf{pk}_2$, but not ssk.

3.2.4 IND-CPA for FHE Pipelines

Given a FHE pipeline $\Pi = (\mathsf{FHE}_1, \mathsf{FHE}_2, \mathsf{Switch})$, we notice that the product scheme $\mathsf{FHE}_1 \times \mathsf{FHE}_2$, enhanced with the ssk, is again an encryption scheme, where the public key is $(\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ssk})$. We can thus consider the IND-CPA game of the pipeline Π, where messages and challenges are encrypted component-wise. We give a formal definition in definition 12.

Definition 12 (IND-CPA security for FHE pipelines)**.** Let $\Pi = (\mathsf{FHE}_1, \mathsf{FHE}_2, \mathsf{Switch})$ be a FHE pipeline. For $b \in \{0,1\}$, the IND-CPA experiment $\mathsf{Exp}_b^{\mathsf{IND\text{-}CPA}}[\mathcal{A}](1^\lambda)$ is defined as follows.

$$\begin{aligned}
&\mathsf{Exp}_b^{\mathsf{IND\text{-}CPA}}[\mathcal{A}](1^\lambda): \\
&\quad (\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{FHE}_1.\mathsf{KeyGen}(1^\lambda), \\
&\quad (\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{FHE}_2.\mathsf{KeyGen}(1^\lambda), \\
&\quad \mathsf{ssk} \leftarrow \mathsf{SwSetup}(\mathsf{sk}_1, \mathsf{pk}_2), \\
&\quad b' \leftarrow \mathcal{A}^{\mathcal{LR}_1^b, \mathcal{LR}_2^b}(1^\lambda, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ssk}), \\
&\quad \textbf{return } b',
\end{aligned}$$

where the oracles $\mathcal{LR}_1^b$ and $\mathcal{LR}_2^b$ are defined by

$$\mathcal{LR}_i^b(m_0, m_1) = \mathsf{FHE}_i.\mathsf{Enc}(\mathsf{pk}_i, m_b).$$

We define the advantage of $\mathcal{A}$ as

$$\mathsf{adv}_\Pi^{\mathsf{IND\text{-}CPA}}[\mathcal{A}](1^\lambda) := \frac{1}{2}\left|\Pr\left[\mathsf{Exp}_0^{\mathsf{IND\text{-}CPA}}[\mathcal{A}](1^\lambda) = 1\right] - \Pr\left[\mathsf{Exp}_1^{\mathsf{IND\text{-}CPA}}[\mathcal{A}](1^\lambda) = 1\right]\right|.$$

We say that the FHE pipeline Π is IND-CPA-secure if

$$\mathsf{adv}_\Pi^{\mathsf{IND\text{-}CPA}}[\mathcal{A}](1^\lambda) \leq \mathsf{negl}(\lambda)$$

for every PPT adversary $\mathcal{A}$.

4 Achieving IND-CPA^{D} Security for Pipelines

In this section we describe how it is possible to achieve the notion of IND-CPA^{D} security for pipelines we described in the previous section.

4.1 Ciphertext Hygiene

We now formally define our key concept of *ciphertext hygiene*, which is a transformation on Switch algorithms. In particular, it enforces that ciphertexts whose underlying plaintext message has uncertainty due to the noise tag t get destroyed. We show next an attack in which the ciphertext hygiene is not enforced, and it has devastating effects on the IND-CPA^{D} security of the pipeline. This is also the blueprint of our attack from Section 5.

Definition 13 (Ciphertext Hygiene)**.** Let FHE_1 and FHE_2 be approximate FHE schemes, and let Switch be an approximately correct scheme switch from FHE_1 to FHE_2. We define $\mathsf{Switch}^{\mathsf{hy}}$ as the scheme switch from FHE_1 to FHE_2 depicted in algorithm 4.

Algorithm 4: The hygienic scheme-switching algorithm $\mathsf{Switch}^{\mathsf{hy}}$

```
Function Switch^hy(ssk, ct)
    ct' ← Switch(ssk, ct)
    ρ ← SwEst_in(ct.t)
    η ← SwEst_out(ct)
    for h ∈ {1, ..., k_2} do
        if ρ_h ≠ 0 then
            ct'_h ← ⊥
        end
    end
    return (ct', η)
```

Ciphertext hygiene is needed to preserve the correspondence between the plaintext labels stored by the security game and the ciphertexts inserted into the target state by the switching oracle. Without this correspondence, the target decryption oracle may be invoked on entries that could not arise in the stand-alone target IND-CPA^{D} game. The following counterexample shows that this is already enough to break the pipeline game.

Malformed target-state entries.

Let FHE_2 be an exact FHE scheme. We say that an entry

$$(x_0, x_1, \mathsf{ct}) \in S_2$$

is *malformed* if

$$\mathsf{Dec}_2(\mathsf{sk}_2, \mathsf{ct}) \notin \{x_0, x_1\}.$$

Such an entry cannot arise in the ordinary $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ game of the target scheme: entries created by encryption or evaluation are labelled by the corresponding plaintext computation, and exact correctness preserves the game-state correspondence.

Lemma 2. *Assume that the target scheme FHE_2 is exact and supports the selector function used below. Consider a pipeline $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ execution in which the switching oracle inserts into the target state an entry*

$$S_2[i] = (x_0, x_1, \mathsf{ct})$$

such that

$$x^\star := \mathsf{Dec}_2(\mathsf{sk}_2, \mathsf{ct}) \notin \{x_0, x_1\}.$$

Then there is an adversary that recovers the challenge bit using one target encryption query, one target evaluation query, and one admissible target decryption query.

Proof. The adversary first queries the target encryption oracle on the pair $(0, 1)$. Let j be the index of the resulting entry, so that

$$S_2[j] = (0, 1, \mathsf{ct}_b),$$

where ct_b encrypts the challenge bit b.

Next, the adversary asks the target evaluation oracle to evaluate the function f on the entries i and j, where

$$f(x, y) = \begin{cases} 0 & \text{if } x \in \{x_0, x_1\}, \\ y & \text{otherwise.} \end{cases}$$

At the level of the game state, the new plaintext labels are

$$f(x_0, 0) = 0 \qquad \text{and} \qquad f(x_1, 1) = 0.$$

Thus the newly created target-state entry is admissible for the restricted decryption oracle.

However, at the ciphertext level, the first input decrypts to $x^\star$, which lies outside $\{x_0, x_1\}$. By exact correctness of FHE_2, the evaluated ciphertext therefore decrypts to

$$f(x^\star, b) = b.$$

A decryption query on the newly created entry reveals the challenge bit. □

Consequently, any non-hygienic switching oracle that creates such a malformed target-state entry with non-negligible probability yields a pipeline that is not $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ secure.

How the counterexample arises from non-hygienic switching.

We now instantiate the above counterexample in the natural setting where the source scheme is approximate and the target scheme is exact. Let FHE_1 be an approximate FHE scheme, let FHE_2 be an exact FHE scheme, and suppose that the switching oracle moves entries from S_1 to S_2 without enforcing ciphertext hygiene.

Let

$$S_1[i] = (m_0, m_1, \mathsf{ct})$$

be an entry in the source state, and let

$$\mathsf{ct}' \leftarrow \mathsf{Switch}(\mathsf{ssk}, \mathsf{ct})$$

be the switched ciphertext. The non-hygienic switching oracle inserts into the target state the entry

$$(\tau(m_0), \tau(m_1), \mathsf{ct}').$$

The switched ciphertext is tied to the actual value carried by the source ciphertext, not necessarily to the plaintext label recorded in the game state. Since FHE_1 is approximate, it may happen that

$$\mathsf{Dec}_1(\mathsf{sk}_1, \mathsf{ct}) \neq m_b.$$

Thus, even if the switching algorithm is correct with respect to the source ciphertext, the target ciphertext may decrypt to

$$\mathsf{Dec}_2(\mathsf{sk}_2, \mathsf{ct}') = \tau(\mathsf{Dec}_1(\mathsf{sk}_1, \mathsf{ct})) \notin \{\tau(m_0), \tau(m_1)\}.$$

In that case, the switching oracle has inserted a malformed entry into S_2.

This attack does not rely on any weakness of the target scheme. The issue is that the target $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ game is being run on a state that its own encryption and evaluation oracles would never generate. Standalone $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ security of FHE_2 therefore gives no protection against such target-state entries.

4.2 Achieving w-IND-CPA^{D} Security from IND-CPA^{D} Components

Our first result is how to achieve $\mathsf{w\text{-}IND\text{-}CPA}^{\mathsf{D}}$ from generic $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ schemes. We require the following strong property of the Switch algorithm, namely that switched ciphertexts are indistinguishable from fresh encryptions of the mapped underlying messages.

Definition 14 (Perfect Re-Encryption)**.** Given a scheme switch Switch from FHE_1 to FHE_2, with plaintext mapping $\tau : \mathcal{M}_1^{k_1} \to \mathcal{M}_2^{k_2}$, we say that it satisfies the *perfect re-encryption* property if for all $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{FHE}_1.\mathsf{KeyGen}(1^\lambda)$, for all $(\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{FHE}_2.\mathsf{KeyGen}(1^\lambda)$, for all $\mathsf{ssk} \leftarrow \mathsf{SwSetup}(\mathsf{sk}_1, \mathsf{pk}_2)$ and for all vectors of ciphertexts $\mathsf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_{k_1})$ under scheme FHE_1 we have that

$$(\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}, \mathsf{Switch}(\mathsf{ssk}, \mathsf{ct})) \approx_s (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}, \widetilde{\mathsf{ct}}),$$

where $\widetilde{\mathsf{ct}} \leftarrow \mathsf{FHE}_2.\mathsf{Enc}(\mathsf{pk}_2, \tau(\mathsf{FHE}_1.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct})))$.

We then show that $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ security of individual schemes compose to $\mathsf{w\text{-}IND\text{-}CPA}^{\mathsf{D}}$ security of the full pipeline, when Switch has perfect re-encryption, and ciphertext hygiene is performed properly.

Theorem 3. *Let* $\Pi = (\mathsf{FHE}_1, \mathsf{FHE}_2, \mathsf{Switch})$ *be a FHE pipeline. If* FHE_1 *and* FHE_2 *are* $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$*-secure FHE schemes for* q_1 *and* q_2 *queries respectively, and* Switch *has the perfect re-encryption property, then the FHE pipeline* $\Pi^{\mathsf{hy}} = (\mathsf{FHE}_1, \mathsf{FHE}_2, \mathsf{Switch}^{\mathsf{hy}})$ *is* (q_1, q_2)-$\mathsf{w\text{-}IND\text{-}CPA}^{\mathsf{D}}$ *secure.*

Proof. Let $\mathcal{A}$ be a PPT adversary against the weak (q_1, q_2)-$\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ game of

$$\Pi^{\mathsf{hy}} = (\mathsf{FHE}_1, \mathsf{FHE}_2, \mathsf{Switch}^{\mathsf{hy}}).$$

For $b \in \{0, 1\}$, let Real_b denote the real weak $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ experiment for Π^{hy} with challenge bit b.

We will compare Real_b to an ideal experiment in which every kept switched ciphertext is replaced by a fresh encryption under FHE_2 of the corresponding stored target label.

The ideal switch experiment. For $b \in \{0, 1\}$, define Ideal_b exactly as Real_b, except that the switch oracle is replaced by the following idealized oracle.

On input

$$J = (j_1, \ldots, j_{k_1}),$$

the oracle forms

$$\mathsf{ct} = (S_1[j_1].\mathsf{ct}, \ldots, S_1[j_{k_1}].\mathsf{ct})$$

and computes the target labels

$$m'_0 := \tau(S_1[j_1].m_0, \ldots, S_1[j_{k_1}].m_0),$$
$$m'_1 := \tau(S_1[j_1].m_1, \ldots, S_1[j_{k_1}].m_1).$$

It also computes the public input-side uncertainty

$$\rho := \mathsf{SwEst}_{\mathsf{in}}(\mathsf{ct}.t).$$

For every component $h \in \{1, \ldots, k_2\}$, if $\rho_h \neq 0$, the oracle sets

$$\widehat{\mathsf{ct}}'_h := \perp .$$

If $\rho_h = 0$, it obtains

$$\mathsf{ct}'_h \leftarrow \mathcal{E}_2^b(m'_{0,h}, m'_{1,h}).$$

Finally, the oracle returns

$$\widehat{\mathsf{ct}}' = (\widehat{\mathsf{ct}}'_1, \ldots, \widehat{\mathsf{ct}}'_{k_2}).$$

Real switch outputs are indistinguishable from ideal switch outputs. We first show that, for every $b \in \{0,1\}$,

$$\mathsf{Real}_b \approx_s \mathsf{Ideal}_b.$$

Consider a switch query in Real_b and let

$$\mathsf{ct}' = \mathsf{Switch}(\mathsf{ssk}, \mathsf{ct})$$

be the switched ciphertext. Fix a kept component h, so that

$$\mathsf{SwEst}_{\mathsf{in}}(\mathsf{ct}.t)_h = 0.$$

Since the $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ game computes tags honestly, by approximate correctness we have the invariant that

$$\mathsf{FHE}_1.\mathsf{Error}(\mathsf{ct}_i, S_1[j_i].m_b, \mathsf{sk}_1) \le \mathsf{ct}_i.t \qquad \text{for every } i \in \{1,\dots,k_1\}.$$

Therefore, by the defining property of $\mathsf{SwEst}_{\mathsf{in}}$,

$$\left\|\tau(\mathsf{FHE}_1.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct}))_h - m'_{b,h}\right\| \le \mathsf{SwEst}_{\mathsf{in}}(\mathsf{ct}.t)_h = 0.$$

Hence

$$\tau(\mathsf{FHE}_1.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct}))_h = m'_{b,h}.$$

By perfect re-encryption,

$$\mathsf{Switch}(\mathsf{ssk}, \mathsf{ct})_h \approx_s \mathsf{FHE}_2.\mathsf{Enc}\left(\mathsf{pk}_2, \tau(\mathsf{FHE}_1.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct}))_h\right).$$

Using the equality above, this gives

$$\mathsf{Switch}(\mathsf{ssk}, \mathsf{ct})_h \approx_s \mathsf{FHE}_2.\mathsf{Enc}(\mathsf{pk}_2, m'_{b,h}).$$

In particular, the perfect re-encryption property implies that $\mathsf{SwEst}_{\mathsf{out}}(\mathsf{ct}) \le B_2$, the fresh encryption bound. Thus, tags assigned to kept components are the same in both experiments, namely B_2, and discarded components are replaced by $\perp$ in both experiments. By an adaptive hybrid over all switch queries and all kept components, we obtain

$$\left|\Pr[\mathsf{Real}_b^{\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Ideal}_b^{\mathcal{A}}(1^\lambda) = 1]\right| \le \mathsf{negl}(\lambda).$$

Separating the two challenge bits. It remains to show that Ideal_0 and Ideal_1 are indistinguishable from the $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ security of the two component schemes.

We define ideal hybrids

$$\mathsf{Hyb}_{\alpha,\beta} \qquad \text{for } \alpha, \beta \in \{0,1\}.$$

In $\mathsf{Hyb}_{\alpha,\beta}$, the FHE_1 oracles are

$$(\mathcal{E}_1^{\alpha}, \mathcal{H}_1, \mathcal{D}_1),$$

and the FHE_2 oracles are

$$(\mathcal{E}_2^{\beta}, \mathcal{H}_2, \mathcal{D}_2).$$

The ideal switch oracle also uses the target challenge bit β, i.e., on a kept component h it uses

$$\mathsf{ct}'_h \leftarrow \mathcal{E}_2^{\beta}(m'_{0,h}, m'_{1,h}).$$

Thus

$$\mathsf{Ideal}_0 = \mathsf{Hyb}_{0,0} \quad \text{and} \quad \mathsf{Ideal}_1 = \mathsf{Hyb}_{1,1}.$$

Reduction to FHE_1. We first compare $\mathsf{Hyb}_{0,0}$ and $\mathsf{Hyb}_{1,0}$. We build an adversary $\mathcal{B}_1$ against the q_1-$\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ security of FHE_1. The adversary $\mathcal{B}_1$ receives pk_1 and oracle access to the source-scheme oracles

$$\mathcal{E}_1^c, \mathcal{H}_1, \mathcal{D}_1$$

for a hidden bit c. It generates

$$(\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{FHE}_2.\mathsf{KeyGen}(1^\lambda)$$

on its own and runs $\mathcal{A}$ on input

$$(1^\lambda, \mathsf{pk}_1, \mathsf{pk}_2).$$

It forwards all source-scheme oracle queries of $\mathcal{A}$ to its own FHE_1 challenger. It simulates all target-scheme queries locally using $\mathsf{sk}_2, \mathsf{pk}_2$ and fixed target bit 0. It still answers switch queries as Ideal_0, i.e. using $\mathcal{E}_2^0$.

If the hidden source bit c of the FHE_1 challenger is 0, the simulated view is distributed as $\mathsf{Hyb}_{0,0}$. If it is 1, the simulated view is distributed as $\mathsf{Hyb}_{1,0}$. Therefore,

$$\frac{1}{2}\left|\Pr[\mathsf{Hyb}_{0,0}^{\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Hyb}_{1,0}^{\mathcal{A}}(1^\lambda) = 1]\right| \leq \mathsf{adv}_{\mathsf{FHE}_1}^{q_1\text{-}\mathsf{IND\text{-}CPA}^{\mathsf{D}}}[\mathcal{B}_1](1^\lambda).$$

Reduction to FHE_2. We now compare $\mathsf{Hyb}_{1,0}$ and $\mathsf{Hyb}_{1,1}$. We build an adversary $\mathcal{B}_2$ against the q_2-$\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ security of FHE_2. The adversary $\mathcal{B}_2$ receives pk_2 and oracle access to

$$\mathcal{E}_2^c, \mathcal{H}_2, \mathcal{D}_2$$

for a hidden bit c. It generates

$$(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{FHE}_1.\mathsf{KeyGen}(1^\lambda)$$

on its own and runs $\mathcal{A}$ on input

$$(1^\lambda, \mathsf{pk}_1, \mathsf{pk}_2).$$

It simulates all source-scheme oracle queries locally using $\mathsf{sk}_1, \mathsf{pk}_1$ and fixed source bit 1. It forwards all ordinary target-scheme oracle queries to its FHE_2 challenger.

For an ideal switch query, $\mathcal{B}_2$ gets the ciphertext from the challenge oracle

$$\mathsf{ct}'_h \leftarrow \mathcal{E}_2^c(m'_{0,h}, m'_{1,h}),$$

and returns it to $\mathcal{A}$.

If the hidden target bit of the FHE_2 challenger is 0, the simulated view is distributed as $\mathsf{Ideal}_{1,0}$. If it is 1, the simulated view is distributed as $\mathsf{Ideal}_{1,1}$. Hence

$$\frac{1}{2}\left|\Pr[\mathsf{Ideal}_{1,0}^{\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Ideal}_{1,1}^{\mathcal{A}}(1^\lambda) = 1]\right| \leq \mathsf{adv}_{\mathsf{FHE}_2}^{q_2\text{-}\mathsf{IND\text{-}CPA}^{\mathsf{D}}}[\mathcal{B}_2](1^\lambda).$$

Combining the two component reductions with the real-to-ideal switch replacement, we get

$$Adv_{\Pi^{\mathsf{hy}}}^{(q_1,q_2)\text{-}\mathsf{w\text{-}IND\text{-}CPA}^{\mathsf{D}}}[\mathcal{A}](1^\lambda) \leq \mathsf{adv}_{\mathsf{FHE}_1}^{q_1\text{-}\mathsf{IND\text{-}CPA}^{\mathsf{D}}}[\mathcal{B}_1](1^\lambda) + \mathsf{adv}_{\mathsf{FHE}_2}^{q_2\text{-}\mathsf{IND\text{-}CPA}^{\mathsf{D}}}[\mathcal{B}_2](1^\lambda) + \mathsf{negl}(\lambda).$$

Since FHE_1 and FHE_2 are respectively q_1- and q_2-$\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ secure, the right-hand side is negligible. Therefore Π^{hy} is weak (q_1, q_2)-$\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ secure.

□ □

4.3 Counterexample to Generic Composition of $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ Schemes

We now show that the analogue of Theorem 3 is false if the scheme-switching key ssk is given to the adversary. The issue is that ssk may arbitrarily contain auxiliary information that is harmless for $\mathsf{IND\text{-}CPA}$, but becomes dangerous in the presence of decryption queries.

Algorithm 5: The modified KeyGen and Dec of the scheme $\mathsf{CKKS}_{\mathsf{seed}}$ and the $\mathsf{SwSetup}$ for the Scheme Switch

Function $\mathsf{KeyGen}_{\mathsf{seed}}(1^\lambda)$
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{CKKS.KeyGen}(1^\lambda)$
  $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
  return $((\mathsf{sk}, \mathsf{seed}), \mathsf{pk})$
Function $\mathsf{Dec}_{\mathsf{seed}}((\mathsf{sk}, \mathsf{seed}), \mathsf{ct})$
  $m \leftarrow \mathsf{CKKS.Dec}(\mathsf{sk}, \mathsf{ct})$
  $\eta \leftarrow \mathsf{PRF}_{\mathsf{seed}}(\mathsf{ct})$
  return $m + \eta$
Function $\mathsf{SwSetup}((\mathsf{sk}, \mathsf{seed}), \mathsf{pk}_2)$
  return seed

Construction of the FHE pipeline Let $\mathsf{CKKS}_{\mathsf{seed}}$ be a noise-flooded variant of the vanilla CKKS algorithm, as described in Algorithm 5. The scheme samples, as part of the secret key, a seed seed used to generate the flooding noise added at decryption time. We assume the flooding distribution is calibrated so that, when seed is kept secret, $\mathsf{CKKS}_{\mathsf{seed}}$ is $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ secure.

Consider now an FHE pipeline

$$(\mathsf{CKKS}_{\mathsf{seed}}, \mathsf{FHE}_2, \mathsf{Switch}),$$

where FHE_2 is any $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$-secure FHE scheme. The switching key produced by $\mathsf{SwSetup}$ is simply the seed seed used for noise flooding in $\mathsf{CKKS}_{\mathsf{seed}}$.

Security analysis As a stand-alone scheme, $\mathsf{CKKS}_{\mathsf{seed}}$ is $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ secure because every decryption output is masked by pseudorandom flooding noise unknown to the adversary. Moreover, the pipeline remains $\mathsf{IND\text{-}CPA}$ secure even when ssk is public: in the $\mathsf{IND\text{-}CPA}$ experiment the adversary only sees ciphertexts and seed, but no decryption outputs. Since seed is sampled independently of the challenge bit and of the encryption randomness, revealing it does not help distinguish challenge ciphertexts.

However, the pipeline is not $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ secure when ssk is public. An adversary receiving $\mathsf{ssk} = \mathsf{seed}$ can query the decryption oracle on admissible ciphertexts of $\mathsf{CKKS}_{\mathsf{seed}}$ and remove the flooding noise by computing

$$\mathsf{Dec}_{\mathsf{seed}}((\mathsf{sk}, \mathsf{seed}), \mathsf{ct}) - \mathsf{PRF}_{\mathsf{seed}}(\mathsf{ct}) = \mathsf{CKKS.Dec}(\mathsf{sk}, \mathsf{ct}),$$

recovering the raw CKKS decryption output. It can then run the standard Li–Micciancio $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ attack against CKKS, using the unmasked decryption values to recover secret-key-dependent error information.

Conclusion Therefore, although the component schemes are individually $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ secure and the pipeline is $\mathsf{IND\text{-}CPA}$ secure with public ssk, the same pipeline is not $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ secure once the adversary is allowed to use ssk together with decryption queries.

Thus, public access to ssk requires extra care. A priori, the stand-alone security of the component schemes gives no guarantee on how secret-key-related auxiliary information, such as ssk, interacts with the decryption oracles in the $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ game. To prove $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ security for the full pipeline, one needs additional conditions controlling this interaction.

4.4 Lifting $\mathsf{IND\text{-}CPA}$ to $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ for pipelines

Our main result is a lifting theorem, i.e. we start from any $\mathsf{IND\text{-}CPA}$-secure pipeline and produce via noise flooding and ciphertext hygiene a full $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$-secure pipeline, where the adversary has access to the ssk. We thus circumvent issues similar to the ones in the counterexample by providing a direct construction where the mechanism with which we achieve $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ is independent of the secret keys and in particular of the ssk. In particular, we give the description of the noise-flooded schemes in Algorithm 6.

We can now prove the following.

Theorem 4. *Let* $\Pi = (\mathsf{FHE}_1, \mathsf{FHE}_2, \mathsf{Switch})$ *be a* $\mathsf{IND\text{-}CPA}$*-secure FHE pipeline. Let* $\mathsf{FHE}_1^{\mathsf{nf}}$ *and* $\mathsf{FHE}_2^{\mathsf{nf}}$ *be constructed as in algorithm 6, where* $\{\mathsf{NF}_{i,t}\}_t$ *denotes a noise-flooding mechanism relative to the message*

Algorithm 6: The modified scheme $\mathsf{FHE}_i^{\mathsf{nf}}$ for $i \in \{1,2\}$

Function $\mathsf{FHE}_i^{\mathsf{nf}}.\mathsf{Enc}(\mathsf{pk}_i, m)$
  $c \leftarrow \mathsf{FHE}_i.\mathsf{Enc}(\mathsf{pk}_i, m)$
  return $\mathsf{ct} = (c, B_i)$ `// fresh tag`

Function $\mathsf{FHE}_i^{\mathsf{nf}}.\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct})$
  $m \leftarrow \mathsf{FHE}_i.\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct})$
  return $\mathsf{NF}_{i,\mathsf{ct}.t}(m)$

Function $\mathsf{FHE}_i^{\mathsf{nf}}.\mathsf{Eval}(\mathsf{pk}_i, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$
  $c \leftarrow \mathsf{FHE}_i.\mathsf{Eval}(\mathsf{pk}_i, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$
  $t \leftarrow \mathsf{Estimate}_i(f, \mathsf{ct}_1.t, \ldots, \mathsf{ct}_k.t)$
  return $\mathsf{ct} = (c, t)$

space $\mathcal{M}_i$ *of* FHE_i*. Let moreover* $\mathsf{Switch}^{\mathsf{nf,hy}}$ *be the same algorithm as* Switch*, where ciphertext hygiene is enforced via* $\mathsf{SwEst}^{\mathsf{nf}}_{\mathsf{out}} = \mathsf{SwEst}_{\mathsf{out}}$ *and*

$$\mathsf{SwEst}^{\mathsf{nf}}_{\mathsf{in}}(t) := \mathsf{SwEst}_{\mathsf{in}}(t + R_1(t)).$$

Then the FHE pipeline $\Pi^{\mathsf{nf}} = (\mathsf{FHE}_1^{\mathsf{nf}}, \mathsf{FHE}_2^{\mathsf{nf}}, \mathsf{Switch}^{\mathsf{nf,hy}})$ *is* $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ *secure.*

Proof. We prove that for every PPT adversary $\mathcal{A}$ making at most q_1 and q_2 queries to the oracles $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively, there exists a PPT adversary $\mathcal{B}$ against the $\mathsf{IND\text{-}CPA}$ game of Π such that

$$\mathsf{adv}^{(q_1,q_2)\text{-}\mathsf{IND\text{-}CPA}^{\mathsf{D}}}_{\Pi^{\mathsf{nf}}}[\mathcal{A}] \leq \mathsf{adv}^{\mathsf{IND\text{-}CPA}}_{\Pi}[\mathcal{B}] + q_1\epsilon_1^{\mathsf{NF}}(\lambda) + q_2\epsilon_2^{\mathsf{NF}}(\lambda).$$

Let $\mathcal{A}$ be a PPT adversary against the (q_1, q_2)-$\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ game of

$$\Pi^{\mathsf{nf}} = (\mathsf{FHE}_1^{\mathsf{nf}}, \mathsf{FHE}_2^{\mathsf{nf}}, \mathsf{Switch}^{\mathsf{nf,hy}}).$$

For $b \in \{0,1\}$, let Real_b denote the real (q_1, q_2)-$\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ experiment for Π^{nf} with challenge bit b.

We define a hybrid experiment Hyb_b that is identical to Real_b, except that the decryption oracles are answered using the stored plaintext labels instead of the actual decryption of the ciphertexts. More precisely, if

$$S_i[j] = (m, m, \mathsf{ct})$$

is an admissible decryption query to $\mathcal{D}_i$, then Hyb_b returns

$$\mathsf{NF}_{i,\mathsf{ct}.t}(m).$$

In contrast, the real experiment returns

$$\mathsf{Dec}_i^{\mathsf{nf}}(\mathsf{sk}_i, \mathsf{ct}) = \mathsf{NF}_{i,\mathsf{ct}.t}(\mathsf{Dec}_i(\mathsf{sk}_i, \mathsf{ct})).$$

We first show that, for every $b \in \{0,1\}$,

$$\left|\Pr[\mathsf{Real}_b^{\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Hyb}_b^{\mathcal{A}}(1^\lambda) = 1]\right| \leq q_1\epsilon_1^{\mathsf{NF}}(\lambda) + q_2\epsilon_2^{\mathsf{NF}}(\lambda).$$

Consider replacing the admissible decryption answers one at a time. Fix one admissible query to $\mathcal{D}_i$ on an entry (m, m, ct) with tag $t = \mathsf{ct}.t$. By tag validity in the simulated game,

$$\|\mathsf{Dec}_i(\mathsf{sk}_i, \mathsf{ct}) - m\| \leq t.$$

Therefore, by the uniform shift-hiding property of the truncated noise-flooding mechanism,

$$\Delta(\mathsf{NF}_{i,t}(\mathsf{Dec}_i(\mathsf{sk}_i, \mathsf{ct})), \mathsf{NF}_{i,t}(m)) \leq \epsilon_i^{\mathsf{NF}}(\lambda).$$

A standard adaptive hybrid argument over at most q_i admissible decryption queries to $\mathcal{D}_i$, for $i \in \{1,2\}$, gives the claimed bound.

It remains to bound the distance between Hyb_0 and Hyb_1 using the $\mathsf{IND\text{-}CPA}$ security of the original pipeline

$$\Pi = (\mathsf{FHE}_1, \mathsf{FHE}_2, \mathsf{Switch}).$$

We construct an adversary $\mathcal{B}$ against the IND-CPA game $\mathsf{Exp}_b^{\mathsf{IND\text{-}CPA}}$ of Π, where b is the challenge bit. The adversary $\mathcal{B}$ receives

$$(1^\lambda, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ssk})$$

and has access to left-right encryption oracles

$$\mathcal{LR}_1^b(m_0, m_1) = \mathsf{FHE}_1.\mathsf{Enc}(\mathsf{pk}_1, m_b), \qquad \mathcal{LR}_2^b(m_0, m_1) = \mathsf{FHE}_2.\mathsf{Enc}(\mathsf{pk}_2, m_b).$$

The reduction $\mathcal{B}$ runs $\mathcal{A}$ on the same input $(\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ssk})$, and simulates the oracles in the hybrid experiment Hyb_b for Π^{nf} to $\mathcal{A}$. The $\mathsf{FHE}_i.\mathsf{Eval}$ algorithms are public given these values, so they are simulated directly by the reduction, which keeps local states $\widehat{S}_1, \widehat{S}_2$. Decryption queries are already simulated and independent of the secret key in Hyb_b. We thus need to simulate encryption and switch queries.

Encryption queries. On a query $\mathcal{E}_i(m_0, m_1)$ from $\mathcal{A}$, the reduction queries its left-right encryption oracle

$$\mathsf{ct} \leftarrow \mathcal{LR}_i^b(m_0, m_1).$$

It assigns to ct the fresh encryption tag used by $\mathsf{FHE}_i^{\mathsf{nf}}$, returns ct to $\mathcal{A}$, and records

$$\widehat{S}_i[\widehat{j}_i] \leftarrow (m_0, m_1, \mathsf{ct}), \qquad \widehat{j}_i \leftarrow \widehat{j}_i + 1.$$

This perfectly simulates the encryption oracle of Π^{nf}, since

$$\mathsf{FHE}_i^{\mathsf{nf}}.\mathsf{Enc} = \mathsf{FHE}_i.\mathsf{Enc}$$

except for the public tag attached to the ciphertext.

Switch queries. On a switch query

$$J = (j_1, \ldots, j_{k_1}),$$

the reduction first checks whether any source entry contains $\perp$. If so, it answers according to the public malformed-query convention.

Otherwise, it forms

$$\mathsf{ct} := (\widehat{S}_1[j_1].\mathsf{ct}, \ldots, \widehat{S}_1[j_{k_1}].\mathsf{ct})$$

and computes the raw switched ciphertext vector

$$\mathsf{ct}' = (\mathsf{ct}'_1, \ldots, \mathsf{ct}'_{k_2}) \leftarrow \mathsf{Switch}(\mathsf{ssk}, \mathsf{ct}).$$

It then applies the public post-processing defining $\mathsf{Switch}^{\mathsf{nf},\mathsf{hy}}$. Let

$$t := (\widehat{S}_1[j_1].\mathsf{ct}.t, \ldots, \widehat{S}_1[j_{k_1}].\mathsf{ct}.t).$$

The reduction computes

$$\rho^{\mathsf{nf}} := \mathsf{SwEst}_{\mathsf{in}}^{\mathsf{nf}}(t) = \mathsf{SwEst}_{\mathsf{in}}(t + R_1(t))$$

and

$$\eta^{\mathsf{nf}} := \mathsf{SwEst}_{\mathsf{out}}^{\mathsf{nf}}(\mathsf{ct}) = \mathsf{SwEst}_{\mathsf{out}}(\mathsf{ct}).$$

It also computes the target plaintext labels

$$m'_0 := \tau(\widehat{S}_1[j_1].m_0, \ldots, \widehat{S}_1[j_{k_1}].m_0),$$

$$m'_1 := \tau(\widehat{S}_1[j_1].m_1, \ldots, \widehat{S}_1[j_{k_1}].m_1).$$

For each component $h \in \{1, \ldots, k_2\}$, it sets

$$\widehat{\mathsf{ct}}'_h := \begin{cases} \mathsf{ct}'_h \text{ with tag } \eta_h^{\mathsf{nf}}, & \text{if } \rho_h^{\mathsf{nf}} = 0, \\ \perp, & \text{otherwise.} \end{cases}$$

It records

$$\widehat{S}_2[\widehat{i}_2] \leftarrow (m'_{0,h}, m'_{1,h}, \widehat{\mathsf{ct}}'_h), \qquad \widehat{i}_2 \leftarrow \widehat{i}_2 + 1,$$

for each h, and returns

$$\widehat{\mathsf{ct}}' = (\widehat{\mathsf{ct}}'_1, \ldots, \widehat{\mathsf{ct}}'_{k_2})$$

to $\mathcal{A}$.

This is exactly the behavior of $\mathsf{Switch}^{\mathsf{nf},\mathsf{hy}}$: it runs the original Switch and then destroys precisely those components whose modified switch estimate is nonzero.

Hence, the reduction $\mathcal{B}$ perfectly simulates Hyb_b to $\mathcal{A}$, where b is the bit in the $\mathsf{IND\text{-}CPA}$ experiment $\mathsf{Exp}_b^{\mathsf{IND\text{-}CPA}}$. Consequently,

$$\frac{1}{2}\left|\Pr[\mathsf{Hyb}_0^{\mathcal{A}}(1^\lambda)=1] - \Pr[\mathsf{Hyb}_1^{\mathcal{A}}(1^\lambda)=1]\right| \le \mathsf{adv}_{\Pi}^{\mathsf{IND\text{-}CPA}}[\mathcal{B}].$$

Combining this bound with the real-to-hybrid transition gives

$$\begin{aligned}\mathsf{adv}_{\Pi^{\mathsf{nf}}}^{(q_1,q_2)\text{-}\mathsf{IND\text{-}CPA}^{\mathsf{D}}}[\mathcal{A}] &= \frac{1}{2}\left|\Pr[\mathsf{Real}_0^{\mathcal{A}}(1^\lambda)=1] - \Pr[\mathsf{Real}_1^{\mathcal{A}}(1^\lambda)=1]\right| \\ &\le \mathsf{adv}_{\Pi}^{\mathsf{IND\text{-}CPA}}[\mathcal{B}] + q_1\epsilon_1^{\mathsf{NF}}(\lambda) + q_2\epsilon_2^{\mathsf{NF}}(\lambda).\end{aligned}$$

The claim follows from the $\mathsf{IND\text{-}CPA}$ security of Π and from the negligible shift-hiding errors of the two noise-flooding mechanisms.

□ □

Remark 5. The theorem assumes $\mathsf{IND\text{-}CPA}$ security of the pipeline. This assumption captures the circular-security issue introduced by publishing sk_1-dependent-material encrypted under pk_2. The lifting theorem does not attempt to prove this circular security. It shows that, once the pipeline is secure without decryption oracles, truncated noise flooding makes the decryption oracles publicly simulatable, even when the adversary is given ssk.

4.5 The Folklore Construction

In this subsection, we revisit the folklore construction of a scheme switch between FHE schemes, i.e. homomorphically evaluating the decryption function of the source scheme within the target scheme, informally

$$\mathsf{Switch}(\mathsf{ct}) = \mathsf{Eval}_2((\tau \circ \mathsf{Dec}_1)(\cdot, \mathsf{ct}), \mathsf{Enc}_2(\mathsf{sk}_1)).$$

We provide here a more formal study of this construction. Recall that we have approximate FHE schemes FHE_1 and FHE_2, each with its own message and ciphertext spaces, $\mathcal{M}_i$ and $\mathcal{C}_i$, as well as a plaintext mapping $\tau : \mathcal{M}_1^{k_1} \to \mathcal{M}_2^{k_2}$.

Suppose that we have a way to encode secret keys of the first scheme in the message space of the second scheme, that is, we are given an invertible map $\phi : \mathcal{SK}_1 \to \mathcal{M}_2^\ell$ for some integer ℓ (e.g. bit decomposition).

We now describe the $\mathsf{SwSetup}^{\mathsf{folk}}$ and $\mathsf{Switch}^{\mathsf{folk}}$ algorithms:

- $\mathsf{SwSetup}^{\mathsf{folk}}(\mathsf{sk}_1, \mathsf{pk}_2)$ computes $e = \phi(\mathsf{sk}_1) \in \mathcal{M}_2^\ell$ and encrypts its components as $c_i \leftarrow \mathsf{Enc}_2(\mathsf{pk}_2, e_i)$. The algorithm returns $\mathsf{ssk} = \{c_i\}_{i\in[\ell]}$.
- $\mathsf{Switch}^{\mathsf{folk}}(\mathsf{ssk}, \mathsf{ct})$ parses $\mathsf{ssk} = \{c_i\}_{i\in[\ell]}$ and $\mathsf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_{k_1})$. Define a function $f_{\mathsf{ct}} : \mathcal{M}_2^\ell \to \mathcal{M}_2^{k_2}$ as follows: given an input $s \in \mathcal{M}_2^\ell$, compute $z_i \leftarrow \mathsf{Dec}_1(\phi^{-1}(s), \mathsf{ct}_i) \in \mathcal{M}_1$ for each $i = 1, \ldots, k_1$, and return $\tau(z_1, \ldots, z_{k_1}) \in \mathcal{M}_2^{k_2}$. Finally, compute $\tilde{\mathsf{ct}} \leftarrow \mathsf{Eval}_2(f_{\mathsf{ct}}, \mathsf{ssk})$ and output $\tilde{\mathsf{ct}} \in \mathcal{C}_2^{k_2}$.

We now analyze the $\mathsf{SwEstimate}$ functions for these algorithms.

Lemma 6. *Let* $\mathsf{Estimate}_2$ *be the evaluation correctness estimation of the approximate FHE scheme* FHE_2*, which is also* B_2*-correct. Suppose that there exists a function* $\omega : \mathbb{R}_{\ge 0}^{k_1} \to \mathbb{R}_{\ge 0}^{k_2}$ *such that* $\|\tau(m_0) - \tau(m_1)\| \le \omega(\|m_0 - m_1\|)$ *for all* $m_0, m_1 \in \mathcal{M}_1^{k_1}$*. Then the functions*

$$\mathsf{SwEst}_{\mathsf{out}}^{\mathsf{folk}}(\mathsf{ct}) = \mathsf{Estimate}_2(f_{\mathsf{ct}}, B_2) \quad \textit{and} \quad \mathsf{SwEst}_{\mathsf{in}}^{\mathsf{folk}}(t) = \omega(t)$$

give a valid $\mathsf{SwEstimate}^{\mathsf{folk}}$ *for the* $\mathsf{Switch}^{\mathsf{folk}}$ *defined above.*

Proof. Let $\mathsf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_{k_1})$ and $m = (m_1, \ldots, m_{k_1})$ be any ciphertext and message pair. Let $t_i = \mathsf{Error}(\mathsf{ct}_i, m_i, \mathsf{sk}_1)$ be the error of ct_i relative to m_i, and write $t = (t_1, \ldots, t_{k_1})$.

Now, since $\mathsf{Switch}(\mathsf{ssk}, \mathsf{ct}) = \mathsf{Eval}_2(f_{\mathsf{ct}}, \mathsf{ssk})$, we have, first by approximate correctness of Eval_2 with the function f_{ct} evaluated on $\phi(\mathsf{sk}_1)$, and then by B_2-correctness, that

$$\begin{aligned}\|\mathsf{FHE}_2.\mathsf{Dec}(\mathsf{sk}_2, \mathsf{Eval}_2(f_{\mathsf{ct}}, \mathsf{ssk})) - f_{\mathsf{ct}}(\phi(\mathsf{sk}_1))\|_2 &\leq \mathsf{Estimate}_2(f_{\mathsf{ct}}, \mathsf{Error}(\mathsf{ssk}, \phi(\mathsf{sk}_1), \mathsf{sk}_2)) \\ &\leq \mathsf{Estimate}_2(f_{\mathsf{ct}}, B_2).\end{aligned}$$

This gives us the bound on $\mathsf{SwEst}^{\mathsf{folk}}_{\mathsf{out}}$.

Regarding $\mathsf{SwEst}^{\mathsf{folk}}_{\mathsf{in}}$, we have that, using the metric properties of τ

$$\begin{aligned}&\|\tau(\mathsf{FHE}_1.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct}_1), \ldots, \mathsf{FHE}_1.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct}_{k_1})) - \tau(m)\|_2 \\ &\qquad \leq \omega\left(\|(\mathsf{FHE}_1.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct}_1), \ldots, \mathsf{FHE}_1.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct}_{k_1})) - m\|_1\right) = \omega(t).\end{aligned}$$

□

Moreover, it is also immediate to see that the folklore construction actually achieves our perfect re-encryption property, under the condition that the destination scheme is circuit private.

Lemma 7. *If* FHE_2 *is a circuit private FHE scheme, then* $\mathsf{Switch}^{\mathsf{folk}}$ *satisfies the perfect re-encryption property.*

Proof. By circuit privacy of FHE_2, we have that $\mathsf{Eval}_2(f, \mathsf{Enc}_2(m)) \approx \mathsf{Enc}_2(f(m))$ for any function f and message m. Applying this with the function f_{ct} defined above and $\mathsf{ssk} = \mathsf{Enc}_2(\phi(\mathsf{sk}_1))$ directly yields

$$\begin{aligned}\mathsf{Switch}^{\mathsf{folk}}(\mathsf{ssk}, \mathsf{ct}) &= \mathsf{Eval}_2(f_{\mathsf{ct}}, \mathsf{ssk}) \\ &\approx \mathsf{Enc}_2(f_{\mathsf{ct}}(\phi(\mathsf{sk}_1))) = \mathsf{Enc}_2(\tau(\mathsf{FHE}_1.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct}_1), \ldots, \mathsf{FHE}_1.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct}_{k_1}))).\end{aligned}$$

□

We then achieve the two following results, which directly follow from Theorems 3 and 4, and Lemmata 6 and 7.

Theorem 8. *Let* FHE_1 *and* FHE_2 *be FHE schemes that are respectively* q_1*- and* q_2*-*$\mathsf{IND\text{-}CPA}^{\mathsf{D}}$*-secure. Assume that* FHE_2 *is circuit private. Let* $\mathsf{Switch}^{\mathsf{folk,hy}}$ *be the folklore scheme switch, with ciphertext hygiene. Then, the FHE pipeline* $(\mathsf{FHE}_1, \mathsf{FHE}_2, \mathsf{Switch}^{\mathsf{folk,hy}})$ *is* $\mathsf{w\text{-}IND\text{-}CPA}^{\mathsf{D}}$*-secure.*

Theorem 9. *Let* $\Pi^{\mathsf{folk}} = (\mathsf{FHE}_1, \mathsf{FHE}_2, \mathsf{Switch}^{\mathsf{folk}})$ *be the folklore pipeline. Assume that* Π^{folk} *is* $\mathsf{IND\text{-}CPA}$*-secure. Then the pipeline* $\Pi^{\mathsf{folk,nf}} = (\mathsf{FHE}_1^{\mathsf{nf}}, \mathsf{FHE}_2^{\mathsf{nf}}, \mathsf{Switch}^{\mathsf{folk,nf,hy}})$ *is* $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$*-secure, where* $\mathsf{Switch}^{\mathsf{folk,nf,hy}}$ *enforces ciphertext hygiene through the function*

$$\mathsf{SwEst}^{\mathsf{folk,nf}}_{\mathsf{in}}(t) = \omega(t + R_1(t)).$$

5 Attacking Non-Hygienic Pipelines

We now present an attack on scheme switching from CKKS to any exact FHE scheme. In particular, we present a $\mathsf{w\text{-}IND\text{-}CPA}^{\mathsf{D}}$ adversary against a FHE pipeline that does not perform ciphertext hygiene, even if the two single FHE schemes are separately $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ secure.

We describe the full attack in Section 5, and in Section 6 we provide concrete implementation and evaluation against the OpenFHE library to give practical evidence of its feasibility.

The objective of this section is to show that ciphertext hygiene is exactly the mechanism through which security is achieved. With this goal in mind, we issue a clear warning to the literature about the insecurity of naive uses of CKKS together with scheme switching. We hope to encourage scheme designers to (i) provide formal proofs that their scheme switch algorithms satisfy $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$-security for FHE pipelines, (ii) supply a $\mathsf{SwEstimate}$ with sufficient precision, and (iii) discard any ciphertext that may carry a plaintext error after the switch.

We illustrate our setting and main attack ideas in Figure 2, while Algorithm 7 lists the main steps of the attack. We closely follow other $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ attacks such as [28, 14, 13].

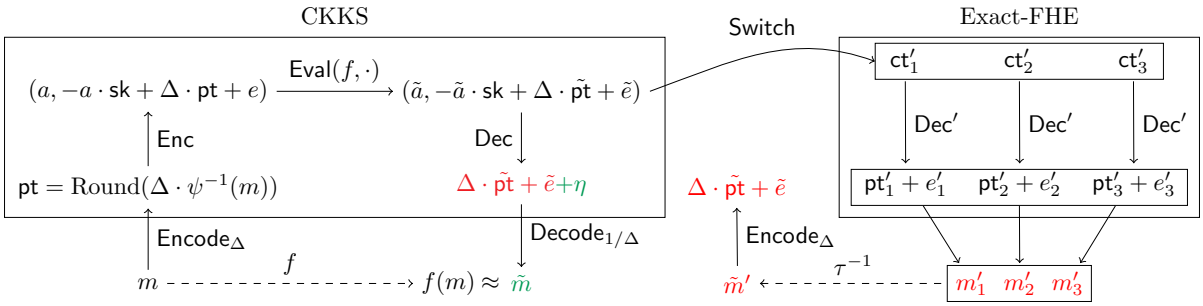

Figure 2: Graphic representation of our w-IND-CPA^{D} attack. In red, the result of the unprotected CKKS decryption. In green, the additional noise added by CKKS decryption as a countermeasure. By using Switch the attacker is able to avoid the countermeasures and to retrieve the sensitive information.

- **Noise amplification in CKKS.** The adversary starts with a ciphertext ct_0 encrypting 0, with a RLWE error e. Through a sequence of scalar multiplications by 2, we make the error grow up to $\|2^k e\|_\infty > \Delta/2$. The ciphertext $\tilde{\mathsf{ct}}$ is now incorrect, and in particular the error e leaks into the plaintext space.
- **Scheme switch to an exact FHE scheme.** The adversary invokes the scheme switch algorithm on $\tilde{\mathsf{ct}}$, producing a new ciphertext ct' under an exact FHE scheme. By correctness of the switching procedure, the plaintext in ct' matches the plaintext in $\tilde{\mathsf{ct}}$, partially including the error e that was previously mixed into the message space. The new ciphertext retains this information, but for the IND-CPA^{D} oracles it is still encrypting zeros.
- **Decryption leakage and key-recovery.** The adversary now queries the decryption oracle on ct'. Since the target scheme does not apply noise flooding, we recover the leakage. By repeating this process across multiple ciphertexts, the adversary accumulates enough information to reconstruct the CKKS secret key sk.

We now proceed to give a detailed description of the attack.

5.1 Setup and Assumptions

Let n be a power of two representing the ring dimension, and let $\mathcal{R} = \mathbb{Z}_q[X]/(X^n + 1)$ be the polynomial ring used by the CKKS scheme. We fix a ciphertext modulus q and a scaling factor Δ, assuming that q is significantly larger than Δ.

Recall that the CKKS encryption algorithm takes as input a vector $z \in \mathbb{C}^{n/2}$ and outputs an RLWE ciphertext via $\mathsf{REnc}(\mathsf{sk}, \mathsf{Encode}_\Delta(z)) \in \mathcal{R}^2$; vice-versa, the decryption procedure takes as input a ciphertext $\mathsf{ct} = (b, a) \in \mathcal{R}^2$ and returns $\mathsf{Decode}_{1/\Delta}(\mathsf{RDec}(\mathsf{sk}, \mathsf{ct}))$.

We assume that the target FHE scheme has plaintext space $\mathbb{Z}_p$, for some modulus p, and that the plaintext mapping τ has the following form:

$$\tau(z_1, \ldots, z_{n/2}) = (\lfloor z_1 \rceil \bmod p, \ldots, \lfloor z_{n/2} \rceil \bmod p)$$

Since the z_i's are complex numbers, we could either consider a mapping that represents $\lfloor z_i \rceil \bmod p$ as the pair $(\lfloor \Re(z_i) \rceil \bmod p, \lfloor \Im(z_i) \rceil \bmod p)$, or we could ignore the imaginary part.

We describe the generic attack assuming that the full information is kept, but in Section 6 we show how to easily adapt this attack in the setting where only the real part is passed to the exact FHE scheme, which is how the OpenFHE library currently implements the scheme-switching functionality.

In our setting, we are assuming that the CKKS scheme is IND-CPA^{D} secure via noise masking, meaning that the decryption algorithm has been transformed into $\mathsf{Decode}_{1/\Delta}(\mathsf{RDec}(\mathsf{sk}, \mathsf{ct}) + \eta)$, where $\eta \in \mathcal{R}$ is a larger polynomial whose entire goal is to completely hide the error term inside ct.

To model the insecure Switch algorithm we consider a scheme switch algorithm designed to match the vanilla CKKS decryption $\mathsf{Decode}_{1/\Delta}(\mathsf{RDec}(\mathsf{sk}, \mathsf{ct}))$. In this way we can obtain a full key recovery. If the Switch algorithm reproduced the IND-CPA^{D} secure masked version of the CKKS decryption $\mathsf{Decode}_{1/\Delta}(\mathsf{RDec}(\mathsf{sk}, \mathsf{ct}) + \eta)$, a similar version of the attack could still be used to break IND-CPA^{D} security of the pipeline (not with a key recovery anymore).

In other words, we model Switch as a perfectly correct switch over the vanilla CKKS decryption.

The adversary now operates in the w-IND-CPA^{D} game for the FHE pipeline $\Pi = (\text{CKKS}, \mathsf{FHE}_2, \mathsf{Switch})$, which is not enforcing ciphertext hygiene.

5.2 Noise amplification and Switch

We start from the all-zero vector $z_0 = (0, \dots, 0) \in \mathbb{C}^{n/2}$, and encrypt it into a ciphertext ct_0. Since $\mathsf{Encode}_\Delta(z_0) = 0 \in \mathcal{R}$, we have that $\mathsf{ct}_0 = (-a \cdot \mathsf{sk} + e, a)$ for some RLWE error e, sampled from the error distribution χ_e. The goal of our attack is to recover e, so that we can subtract it and easily retrieve the secret key sk.

Our attack depends on the choice of an integer k, representing how many times we double the ciphertext ct_0; we expect k to be around $\log \Delta$, but we precisely analyze how this impacts the effectiveness of the attack in Section 5.4.

Once we have chosen k, we can compute the ciphertexts $\mathsf{ct}_{i+1} = \mathsf{Eval}(\text{Add}, \{\mathsf{ct}_i, \mathsf{ct}_i\})$ until we get to ct_k. In particular, this ciphertext is of the form

$$\mathsf{ct}_k = (-a' \cdot \mathsf{sk} + 2^k e, a').$$

If we would decrypt this CKKS ciphertext, we would get the complex number $z^{(k)} = \psi\left(\frac{2^k e}{\Delta}\right)$; by applying ψ^{-1} and rescaling, we would easily be able to obtain the original error e and be done with the attack. This is what the CKKS attack by Li and Micciancio [28] did.

In our case, we assume that the CKKS scheme has been protected against this type of attacks via noise masking; however, we are able to get information about $z^{(k)}$ via the scheme-switching functionality.

By assuming that the scheme switch algorithm is correct with respect to the vanilla CKKS decryption, we get that

$$\tau\left(\mathsf{Decode}_{1/\Delta}(\mathsf{RDec}(\mathsf{sk}, \mathsf{ct}))\right) = \mathsf{Dec}'(\mathsf{sk}', \mathsf{Switch}(\mathsf{ssk}, \mathsf{ct}))$$

for any CKKS ciphertext ct.

In particular, we can publicly compute $\mathsf{ct}' = \mathsf{Switch}(\mathsf{ssk}, \mathsf{ct}_k)$. Notice that, while playing the IND-CPA^{D} game, all the ciphertexts ct_i have corresponding plaintext message z_0, since in the plaintext space we should just be computing $2^i \cdot 0 = 0$; this means that also the switched ciphertext $\tau(z_0)$ is associated to the plaintext vector of all zeros.

This means that we are allowed to ask the decryption of ct' to the decryption oracle of the exact FHE scheme; in particular, ct' is a vector of length n (or $n/2$ if we drop imaginary parts) of ciphertexts of the target FHE scheme, and by calling the decryption oracle we get a vector of plaintext messages $(m_1, \dots, m_n) \in \mathbb{Z}_p^n$.

Now, given that $z^{(k)} = \psi\left(\frac{2^k e}{\Delta}\right)$ would be the vanilla CKKS decryption of ct_k, given our assumption about Switch, we get that

$$\tau(z^{(k)}) = (m_1, \dots, m_n).$$

If we assume that $\|z^{(k)}\|_\infty < p/2$, then the reduction modulo p does not happen in the message space of the exact FHE scheme, and thus we can reconstruct an approximation $\tilde{z}$ by pairing up the approximated real and imaginary part from the m_i's:

$$\tilde{z}_i = \lfloor \Re(z_i^{(k)}) \rceil + i \cdot \lfloor \Im(z_i^{(k)}) \rceil$$

Since we are rounding to the closest integer, we get that $|\tilde{z}_i - z_i^{(k)}| \leq |\Re(z_i^{(k)}) - \lfloor \Re(z_i^{(k)}) \rceil| + |\Im(z_i^{(k)}) - \lfloor \Im(z_i^{(k)}) \rceil| \leq 1$. Thus we have

$$\left\| \tilde{z} - \psi\left(\frac{2^k e}{\Delta}\right) \right\|_\infty \leq 1.$$

We call the vector $\tilde{z}$ the *leakage*, because the IND-CPA^{D} oracles expect it to be 0, but we managed to get information about the RLWE error e through the noise amplification step and then Switch and decrypting.

5.3 Re-encoding the leakage and key-recovery

The main idea of the key-recovery is now that we have an approximation $\tilde{z}$ of the value $\psi\left(\frac{2^k e}{\Delta}\right)$, from which we hope to reconstruct information about the error e.

The attack is now very simple: we compute $\tilde{e} = \mathsf{Encode}_\Delta(\tilde{z})$, an encoding of our leakage, and then round down $\tilde{e}$, obtaining a guess for the original RLWE error

$$e' = \left\lfloor \frac{\mathsf{Encode}_\Delta(\tilde{z})}{2^k} \right\rceil.$$

Once we have this e', we go back to the original ciphertext $\mathsf{ct}_0 = (b, a) = (-a \cdot \mathsf{sk} + e, a)$ and compute $\mathsf{sk}' = (e' - b) \cdot a^{-1}$.

In the rest of this section we show that, for appropriate values of k, we actually compute $e' = e$ and thus $\mathsf{sk}' = \mathsf{sk}$.

Recall that we have the following relation

$$\psi\left(\frac{2^k e}{\Delta}\right) = \tilde{z} + \varepsilon,$$

where $\varepsilon \in \mathbb{C}^{n/2}$ is a small approximation error, in particular with $\|\varepsilon\|_\infty \leq 1$.

Since ψ is a linear isomorphism between $\mathbb{R}[X]/(X^n + 1)$ and $\mathbb{C}^{n/2}$, we can apply ψ^{-1}, multiply by Δ and obtain that

$$2^k e = \Delta\psi^{-1}(\tilde{z}) + \Delta\psi^{-1}(\varepsilon).$$

Notice that the encoding algorithm of CKKS gives us that $\tilde{e} = \mathsf{Encode}_\Delta(\tilde{z}) = \mathrm{Round}(\Delta\psi^{-1}(\tilde{z})) \in \mathcal{R}$, so $\|\tilde{e} - \Delta\psi^{-1}(\tilde{z})\|_\infty \leq 1/2$.

By denoting as $\tilde{\varepsilon}$ the difference and dividing by 2^k we get that

$$e = \frac{\tilde{e} + \tilde{\varepsilon} + \Delta\psi^{-1}(\varepsilon)}{2^k},$$

so in order to recover the correct e we need to ensure that $\left\|\tilde{\varepsilon} + \Delta\psi^{-1}(\varepsilon)\right\|_\infty < 2^{k-1}$. Since $\tilde{\varepsilon}$ is a rounding error, we have that $\|\tilde{\varepsilon}\|_\infty \leq 1$, so we can approximate our requirement to $\left\|\Delta\psi^{-1}(\varepsilon)\right\|_\infty < 2^{k-1}$.

Algorithm 7: CKKS → Exact-FHE Key-Recovery Attack

Input: Ring dimension n (power of two); scaling factor Δ; ciphertext modulus q; oracle access to all $\mathcal{E}_i^b, \mathcal{H}_i, \mathcal{D}_i$ oracles, and the switch oracle $\mathcal{SW}$

Output: CKKS secret key sk

Procedure RecoverSK*()*

Choose an integer k according to section 5.4;
$\mathsf{ct}_0 \leftarrow \mathcal{E}_1^b(0, 0)$;
for $i \leftarrow 1$ **to** k **do**
  $\mathsf{ct}_i \leftarrow \mathcal{H}_1(\mathsf{Add}, \{i-1, i-1\})$
end
$\mathsf{ct}' \leftarrow \mathcal{SW}(k)$;
$\mathbf{m}' \leftarrow \mathcal{D}_2(0), \ldots, \mathcal{D}_2(n-1)$ `// oracle leak:` $\tau(z^{(k)})$;
Reconstruct $\tilde{z}_i \leftarrow \lfloor \Re(m_i') \rceil + i \cdot \lfloor \Im(m_i') \rceil$;
$\tilde{e} \leftarrow \mathsf{Encode}_\Delta(\tilde{z})$;
$e' \leftarrow \lfloor \tilde{e}/2^k \rceil$;
Decompose ct_0 as (b, a);
Compute $\mathsf{sk}' \leftarrow (e' - b) \cdot a^{-1} \bmod q$;
return sk';

5.4 Choosing the correct k

Recall that our attack correctly finds the original error e whenever the following two conditions are met:

$$\left\|\psi\left(\frac{2^k e}{\Delta}\right)\right\|_\infty < p/2, \text{ and } \left\|\Delta\psi^{-1}(\varepsilon)\right\|_\infty < 2^{k-1},$$

where e is the original RLWE error that we are trying to recover, ε is a vector in $\mathbb{C}^{n/2}$ with $\|\varepsilon\|_\infty \leq 1$, and p is the plaintext modulus of the exact FHE scheme.

The first equation requires that no modular wrap around exists, so that our leakage $\tilde{z}$ is actually a good approximation of the leaked error; we are requiring k to be *not too big*. The second equation is instead required to correctly decode the error from our re-encoded guess $\tilde{e}$, so we are asking k to be *not too small*, otherwise we do not get any leakage.

We analyze the required intervals for k and find that there usually is a *sweet spot* where both conditions hold, allowing us to make the attack work. We make use of the following result on sub-gaussian distributions from [35]

Theorem 10. *Let Z be a uniform random variable on the cube $\|Z\|_\infty \leq B$ in $\mathbb{C}^N$, and let $A \in \mathbb{C}^{N\times N}$ be a matrix with $|A_{i,j}|_2 = 1$. Then $\Pr[\|A \cdot Z\|_\infty \geq t] \leq 2N \exp(-\frac{3t^2}{2NB^2})$. As a consequence,*

$$\mathbb{E}\left[\|A \cdot Z\|_\infty\right] = B\Theta\left(\sqrt{N \log N}\right)$$

We then easily get the following two corollaries:

1. Starting from a vector $z \in \mathbb{C}^{n/2}$ with $\|z\|_\infty \leq 1$, then $\mathbb{E}\left[\|\psi^{-1}(z)\|_\infty\right] = \Theta\left(\sqrt{\frac{\log n}{n}}\right)$
2. Starting from a polynomial $e \in \mathcal{R}$, then $\mathbb{E}\left[\|\psi(e)\|_\infty\right] = \Theta\left(\sqrt{n \log n}\right) \cdot \|e\|_\infty$

This means that, on average, we can expect that

$$\left\|\psi\left(\frac{2^k e}{\Delta}\right)\right\|_\infty \approx \frac{2^k}{\Delta} \cdot \|e\|_\infty \cdot \sqrt{n \log n},$$

and on the other end

$$\left\|\Delta\psi^{-1}(\varepsilon)\right\|_\infty \approx \Delta\sqrt{\frac{\log n}{n}}.$$

Taking logarithms, and ignoring additive constants, we obtain the approximate conditions

$$k - \log\Delta + \log\|e\|_\infty + \frac{1}{2}\log n + \frac{1}{2}\log\log n \leq \log p,$$
$$\log\Delta - \frac{1}{2}\log n + \frac{1}{2}\log\log n \leq k.$$

Equivalently, denoting

$$\mu = \log\Delta - \frac{1}{2}\log n,$$

we expect the attack to succeed for

$$\mu + \frac{1}{2}\log\log n \leq k \leq \mu - \frac{1}{2}\log\log n + \log p - \log\|e\|_\infty.$$

Thus the width of the admissible k-window is approximately

$$W = \log p - \log\|e\|_\infty - \log\log n.$$

In particular, the scale Δ shifts the interval, but does not affect its width.

For example, working on the ring of dimension $n = 2^{12}$, with scaling factor $\Delta = 2^{50}$, we have $\mu = 46$; in this setting we usually get $\log\|e\|_\infty \approx 10$. If we use a plaintext modulus for the exact FHE scheme of $p = 2^{17}$, the allowed interval becomes $47.8 < k < 51.2$. We show extended experimental results about the size of $\|e\|_\infty$ and the allowed ranges for k in section 6.4, where we evaluate concrete OpenFHE implementations.

6 Proof-of-Concept Implementation

We demonstrate the feasibility of our generic attack from Section 5 by implementing it using the OpenFHE library [3].

We selected this library since it is the only actively maintained major FHE library that natively implements scheme-switching functionalities.

In this section we describe OpenFHE's scheme switch feature, explain how to adapt the generic attack to this setting, and conduct an evaluation of the attack on different parameter sets.

6.1 OpenFHE Scheme Switch

OpenFHE introduced scheme-switching capabilities between CKKS and FHEW, allowing users to switch schemes in both directions; we focus only on the CKKS-to-FHEW direction.

Recall that FHEW [21] is a LWE-based FHE scheme, meaning that the plaintext space is $\mathbb{Z}_p$, for some plaintext modulus p, and ciphertexts are pairs $(b, \mathbf{a})$ with $b \in \mathbb{Z}_q$ and $\mathbf{a} \in \mathbb{Z}_q^\ell$, where q is the ciphertext modulus, and ℓ the LWE dimension.

To instantiate CKKS we need to specify a ring dimension n, a power of two, and a scaling factor Δ; the library automatically computes the needed ciphertext modulus. Notice that we can then pack up to $n/2$ complex values in a single CKKS ciphertext; in particular, for any smaller power of two $n' \leq n/2$, we can create packed messages by embedding $\iota : \mathbb{C}^{n'} \to \mathbb{C}^{n/2}$.

When switching scheme, the corresponding ciphertext mapping is given by

$$\tau(z_1, \ldots, z_{n'}) = (\lfloor \Re z_1 \rceil \bmod p, \ldots, \lfloor \Re z_{n'} \rceil \bmod p)$$

The library thus only allows us to see the real part of the output, since in the most common use cases all of the z_i are actually real numbers. In the next subsection we show that this partial knowledge requires us to execute the attack on many different ciphertexts, getting some information about the original error from each one of them.

In particular, we choose $n' = n/2$, i.e. a *fully packed* vector, and ask for the decryption of two CKKS ciphertexts, resulting in a total of n FHEW decrypted messages. Since each of them gives us a linear relation on the secret key, we recover it by solving a linear system via Gaussian elimination, having complexity at most $O(n^3)$.

6.2 Using Real-Only Leakage

Recall that, in the generic attack, we had $z^{(k)} = \psi\left(\frac{2^k e}{\Delta}\right)$, and the decryptions gave us $\lfloor \Re(z_i^{(k)}) \rceil$ and $\lfloor \Im(z_i^{(k)}) \rceil$. In our case, when switching scheme to FHEW, OpenFHE only reveals to us the values $\lfloor \Re(z_i^{(k)}) \rceil$.

6.2.1 The Real Part of ψ

We are now going to analyze how the canonical isomorphism $\psi : \mathbb{R}[X]/(X^n + 1) \to \mathbb{C}^{n/2}$ behaves on certain classes of polynomials. Denote by $R = \mathbb{R}[X]/(X^n + 1)$, and given any $r \in R$ let r_i be its i-th coefficient.

Consider the following two spaces

$$R^- = \{r \in R : r_{n-i} = -r_i \ \forall i = 1, \ldots, n/2\},$$

and

$$R^+ = \{r \in R : r_0 = 0, r_{n-i} = r_i \ \forall i = 1, \ldots, n/2\}.$$

Notice that any polynomial r can uniquely be decomposed into a sum of $r^+ \in R^+$ and $r^- \in R^-$. In particular, $2r^+(X) = r(X) - r(1/X)$ and $2r^-(X) = r(X) + r(1/X)$. Indeed, since $X^n = -1$, we have that $1/X^i = -X^{n-i}$; thus, if $r(X) = r_0 + r_1 X + \cdots + r_{n-1} X^{n-1}$, then

$$r(1/X) = r_0 - r_1 X^{n-1} - \cdots - r_{n-1} X.$$

Now, when computing the isomorphism ψ, we evaluate the polynomial at $2n$-th roots of unity ζ. It is easy to see that

$$\psi(R^-) = \mathbb{R}^{n/2}, \ \psi(R^+) = i \cdot \mathbb{R}^{n/2}$$

So the splitting of $R = R^- \oplus R^+$ is mapped by ψ into the splitting of $\mathbb{C}^{n/2} = \mathbb{R}^{n/2} \oplus i \cdot \mathbb{R}^{n/2}$.

6.2.2 Real-Only Step

Going back to our setting, we have that $z^{(k)} = \psi\left(\frac{2^k e}{\Delta}\right)$. Now, consider the splitting of the RLWE error as $e = e^+ + e^-$; then $\psi(e) = \psi(e^+) + \psi(e^-)$, and we know that $\psi(e^+)$ is totally imaginary, and $\psi(e^-)$ is totally real.

In particular, we have that $\Re(\psi(e)) = \psi(e^-)$, meaning that

$$\Re(z^{(k)}) = \psi\left(\frac{2^k e^-}{\Delta}\right).$$

Thus, when we re-encode the leakage $\lfloor \Re(z_i^{(k)}) \rceil$ we obtain *exactly* the value e^-, like in the generic attack, assuming that our value of k is big enough.

6.2.3 Linear Step

At this point we have fully recovered the polynomial $e^- \in \mathcal{R}$; this gives us only partial information about the original RLWE error e. However, we can repeat this procedure many times, starting with many different es and obtaining their *folded* version e^-s. In particular, if $e(X) = e_0 + e_1 X + \cdots + e_{n-1}X^{n-1}$, then $e^-(X) = \frac{1}{2}\left(e(X) + e(1/X)\right)$, which is given by

$$e_0 + X\frac{e_1 - e_{n-1}}{2} + X^2\frac{e_2 - e_{n-2}}{2} + \dots$$

Instead of trying to recover all the coefficients of the original error, we use the partial information to create linear equations involving the secret key sk of CKKS. Recall indeed that, given a ciphertext $\mathsf{ct} = \mathsf{Enc}(\mathsf{sk}, 0) = (b, a)$ on $\mathcal{R}^2$, we can express the relations in the ciphertext as a linear system $b = -A \cdot \mathsf{sk} + e$, where A is the n-by-n circulant matrix associated to a, and we consider b, e, sk as vectors of coefficients.

Notice now that, denoting by A_i the i-th row of A, we have that $e_i = b_i + \langle A_i, \mathsf{sk}\rangle$. Using our knowledge of the recovered coefficients of e^-, i.e. $e_i^- = \frac{e_i - e_{n-i}}{2}$, we can write the following linear equations for all indices $i = 1, \dots, n/2$:

$$2e_i^- = b_i - b_{n-i} + \langle A_i - A_{n-i}, \mathsf{sk}\rangle.$$

Since any switched-and-decrypted ciphertext gives us $n/2$ linear equations, and sk is a vector of length n, we only need to recover the e^- value of two ciphertexts.

6.3 Responsible Disclosure

We responsibly disclosed our findings to the OpenFHE development team prior to the public release of the paper. We shared an earlier version of the manuscript and a proof-of-concept implementation for key-recovery in CKKS-to-FHEW scheme switching under standard parameters.

The OpenFHE team was responsive and provided detailed feedback. In particular, they emphasized that CKKS-to-FHEW scheme switching is an *experimental* capability in OpenFHE, and that the OpenFHE documentation does not claim $\mathsf{IND\text{-}CPA^D}$ security for scheme-switching functionalities (in contrast to the security claims for standalone CKKS). Therefore our results do not break any claim from the OpenFHE team.

We are grateful to the OpenFHE team for their cooperation and for the constructive technical discussion, and we have incorporated this feedback into the manuscript.

6.4 Experimental Results

We conducted a series of experiments to evaluate the effectiveness of our attack across different CKKS parameters.

We used OpenFHE version `1.3.0` and instantiated both CKKS and FHEW with parameters targeting 128-bit security, following the recommendations of the Homomorphic Encryption Standard [1]. All experiments were conducted on AMD EPYC 9374F processors.

Our code is available at `https://github.com/drago-96/FHE_Switch`.

For many parameter configurations, we attempted to reconstruct the original CKKS error term from the output of the ciphertext after Switch, following the methodology described in this section. Our evaluation focuses on measuring the infinity norm of the difference between the reconstructed error polynomial and the original CKKS error.

Table 1: Measured and estimated CKKS encryption error sizes and the resulting predicted length of the k-window.

$\log n$	$\log\|e\|_\infty$ measured	$\log\|e\|_\infty$ estimated	W estimated
12	9.81	9.91	3.61
13	10.36	10.47	2.94
14	10.89	11.02	2.30
15	11.43	11.57	1.67

6.4.1 Encryption Error Size

First of all, we study the size $\|e\|_\infty$ of the initial CKKS encryption, giving an a priori estimate from the internals of OpenFHE, and validating it empirically.

In a CKKS public-key encryption of zero, the decrypted error is roughly of the form

$$e \approx e_1 + se_2 + ue_{\mathsf{pk}},$$

where the error terms are sampled from the discrete Gaussian used by OpenFHE, with standard deviation σ, and s, u are ternary. Approximating the convolution terms as independent Gaussian sums, each coefficient has standard deviation $\sigma_e \approx \sigma\sqrt{1+\frac{4n}{3}}$, hence

$$\|e\|_\infty \approx \sigma\sqrt{1+\frac{4n}{3}} \cdot \sqrt{2\ln n}.$$

Equivalently, for large n,

$$\log\|e\|_\infty \approx \log\sigma + \frac{1}{2}\log n + \frac{1}{2}\log\log n.$$

Substituting this into the window-width formula from Section 5.4 gives the heuristic

$$W \approx \log p - \log\sigma - \frac{1}{2}\log n - \frac{3}{2}\log\log n.$$

For OpenFHE's default CKKS error parameter, $\sigma \approx 3.2$, and for the default FHEW plaintext modulus $p = 2^{17}$, this predicts a positive k-window for the ring dimensions used in our experiments. We checked this directly by sampling ten CKKS encryptions of zero for each ring dimension, using $\Delta = 2^{50}$. The results are shown in Table 1. Thus, for $\log p = 17$, the predicted k-window is non-empty for all tested ring dimensions up to $\log n = 15$. The estimate also predicts a positive, though narrower, window at $\log n = 16$.

6.4.2 Finding the k-window

In this experiment, we investigate the regimes of parameters in which our attack succeeds, i.e. how well we reconstruct the original RLWE error. In particular, we validate the heuristic estimate of the possible window for k from section 5.4

$$\mu + \frac{1}{2}\log\log n \le k \le \mu - \frac{1}{2}\log\log n + \log p - \log\|e\|_\infty.$$

More concretely, we fix the CKKS ring dimension to $n = 2^{12}$, the LWE plaintext modulus to $p = 2^{17}$, and then for different choices of scaling factors Δ and amplification factors k we run the attack to compute the reconstructed RLWE error

$$e' = \left\lfloor \frac{\mathsf{Encode}_\Delta(\tilde{z})}{2^k} \right\rceil.$$

Table 2 then reports in each cell the value $\log\|e - e'\|_\infty$, where e is the original RLWE error of the CKKS encryption. Based on its magnitude, we distinguish two scenarios:

Exact recovery. The attack fully reconstructs the original error. In this case, the adversary obtains exact linear equations in the CKKS secret key, enabling full key recovery via standard linear algebra. We discuss this case in the next subsection.

Table 2: Heatmap showing the effectiveness of the attack in section 6, for parameters $p = 2^{17}$ and $n = 2^{12}$. The horizontal axis is the amplification factor k, and the vertical axis is the logarithm of the CKKS scaling factor Δ. A red empty cell denotes exact error recovery, while an orange cell denotes residual error below 2^4. Cells underlined with a thick rule lie inside the predicted k-window.

$\log \Delta / k$	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
30	4	3.1	2.1	1.1							9.4	10	10	10	10	10	10	10	10	10	10	10
32	6.2	5.3	4.2	3.1	2.1	1						4.7	9.5	10	10	10	10	10	10	10	10	10
34	8.1	7.1	6.1	5.1	4.2	3.1	2.1	1							9.4	10	10	10	10	10	10	10
36	9.9	9	8.1	7.2	6	5	4	3.1	2.1	1							9.3	10	10	10	10	10
38	10	10	9.8	9	8.1	7.1	6.1	5.1	4.1	3.2	2.1	1							9.5	10	10	10
40	10	10	10	10	9.9	9.1	8	7.1	6.1	5.1	4.1	3	2.1	1							9.3	10
42	10	10	10	10	10	10	9.9	9.1	8.3	7.1	6.4	5.1	4	3.1	2.1	1						4.7
44	10	10	10	10	10	10	10	10	9.8	9.1	8	7	6	5	4.1	3.1	2	1				
46	10	10	10	10	10	10	10	10	10	10	9.8	9.2	8	7	6	5.1	4.1	3.3	2.1	1.3		
48	10	10	10	10	10	10	10	10	10	10	10	10	9.8	9.1	8	7.2	6.2	5.2	4.1	3.1	2.2	1

$\log \Delta / k$	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
40	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
42	9.5	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
44			9.4	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
46					9.4	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
48							9.4	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
50	2.1	1						9.4	10	10	10	10	10	10	10	10	10	10	10	10	10	10
52	4	3.1	2.1	1							9.3	10	10	10	10	10	10	10	10	10	10	10
54	6.1	5.1	4.1	3	2.1	1							9.2	10	10	10	10	10	10	10	10	10
56	8.2	7.1	6.1	5.1	4.1	3.1	2.1	1.1							9.4	10	10	10	10	10	10	10
58	9.8	9.2	8.1	7.1	6.2	5.1	4.2	3.1	2.3	1.1						4.7	9.5	10	10	10	10	10

Partial recovery. Although the error is not recovered exactly, the approximation is close enough to leak secret-dependent information. This leakage can significantly reduce the effective security level of the scheme. For example, it may enable lattice attacks that are theoretically polynomial-time, but currently lack optimized implementations.

6.4.3 Executing the Full Attack

Once experiments confirmed our understanding of the effects of the choice of k, we can execute our full attack. We implemented the $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ game for the CKKS-to-FHEW pipeline using OpenFHE bindings, and implemented an adversary executing Algorithm 7 to recover the secret key of the CKKS scheme. We used the MAGMA [6] software to solve the resulting linear system in the secret key.

We executed the attack for CKKS ring dimensions $n = 2^{12}, 2^{13}, 2^{14}$, using the standard values from the OpenFHE example $\Delta = 2^{52}$, $p = 2^{17}$, choosing the most appropriate value of k for each case, which ended up being $k = 49$ for all the three cases. We report the results of this experiment in Table 3, in particular the running time and the memory consumption of MAGMA solving the linear system; other attack costs are negligible. We extrapolate the resources required for running the attack with $n = 2^{15}$.

Table 3: Computational resources needed for the final key-recovery step in our attack, measured as the time T and memory M required to solve the linear system in the real-only case. Results are shown as a function of the CKKS ring dimension n. For $n = 2^{15}$ we linearly interpolate the results from the smaller dimension experiments.

$\log n$	12	13	14	15
T	1m58s	10m23s	1h0m27s	$\approx$ 5h30m
M	6.3GB	25.3GB	100.4GB	$\approx$ 400GB

References

[1] Albrecht, M., Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Halevi, S., Hoffstein, J., Laine, K., Lauter, K., Lokam, S., Micciancio, D., Moody, D., Morrison, T., Sahai, A., Vaikun-

tanathan, V.: Homomorphic encryption standard. Cryptology ePrint Archive, Report 2019/939 (2019), https://eprint.iacr.org/2019/939

[2] Alexandru, A., Badawi, A.A., Micciancio, D., Polyakov, Y.: Application-aware approximate homomorphic encryption: Configuring FHE for practical use. Cryptology ePrint Archive, Report 2024/203 (2024), https://eprint.iacr.org/2024/203

[3] Badawi, A.A., Alexandru, A., Bates, J., Bergamaschi, F., Cousins, D.B., Erabelli, S., Genise, N., Halevi, S., Hunt, H., Kim, A., Lee, Y., Liu, Z., Micciancio, D., Pascoe, C., Polyakov, Y., Quah, I., R. V., S., Rohloff, K., Saylor, J., Suponitsky, D., Triplett, M., Vaikuntanathan, V., Zucca, V.: OpenFHE: Open-source fully homomorphic encryption library. Cryptology ePrint Archive, Report 2022/915 (2022), https://eprint.iacr.org/2022/915

[4] Bernard, O., Joye, M., Smart, N.P., Walter, M.: Drifting towards better error probabilities in fully homomorphic encryption schemes. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part VIII. LNCS, vol. 15608, pp. 181–211. Springer, Cham (May 2025). 10.1007/978-3-031-91101-9_7

[5] Bian, S., Zhao, Z., Zhang, Z., Mao, R., Suenaga, K., Jin, Y., Guan, Z., Liu, J.: HEIR: A unified representation for cross-scheme compilation of fully homomorphic computation. In: NDSS 2024. The Internet Society (Feb / Mar 2024)

[6] Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. J. Symbolic Comput. **24**(3-4), 235–265 (1997). 10.1006/jsco.1996.0125, http://dx.doi.org/10.1006/jsco.1996.0125, computational algebra and number theory (London, 1993)

[7] Bossuat, J.P., Costache, A., Mouchet, C., Nürnberger, L., Troncoso-Pastoriza, J.R.: Practical q-ind-cpa-d-secure approximate homomorphic encryption. Cryptology ePrint Archive (2024)

[8] Boura, C., Gama, N., Georgieva, M., Jetchev, D.: Chimera: Combining ring-lwe-based fully homomorphic encryption schemes. Journal of Mathematical Cryptology **14**(1), 316–338 (2020)

[9] Brakerski, Z.: When homomorphism becomes a liability. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 143–161. Springer, Berlin, Heidelberg (Mar 2013). 10.1007/978-3-642-36594-2_9

[10] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS 2012. pp. 309–325. ACM (Jan 2012). 10.1145/2090236.2090262

[11] Cao, G., Chatel, S., Knabenhans, C.: HE-based on-the-fly MPC, revisited: Universal composability, approximate and imperfect computation, circuit privacy. Cryptology ePrint Archive, Report 2025/1845 (2025), https://eprint.iacr.org/2025/1845

[12] Checri, M., Clet, P.E., Renard, M., Sirdey, R.: Impossibility of CPAD security for a class of FHE schemes. Cryptology ePrint Archive, Paper 2026/203 (2026), https://eprint.iacr.org/2026/203

[13] Checri, M., Sirdey, R., Boudguiga, A., Bultel, J.P.: On the practical CPA^D security of "exact" and threshold FHE schemes and libraries. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part III. LNCS, vol. 14922, pp. 3–33. Springer, Cham (Aug 2024). 10.1007/978-3-031-68382-4_1

[14] Cheon, J.H., Choe, H., Passelègue, A., Stehlé, D., Suvanto, E.: Attacks against the IND-CPA^{D} security of exact FHE schemes. In: Luo, B., Liao, X., Xu, J., Kirda, E., Lie, D. (eds.) ACM CCS 2024. pp. 2505–2519. ACM Press (Oct 2024). 10.1145/3658644.3690341

[15] Cheon, J.H., Hong, S., Kim, D.: Remark on the security of CKKS scheme in practice. Cryptology ePrint Archive, Report 2020/1581 (2020), https://eprint.iacr.org/2020/1581

[16] Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 409–437. Springer, Cham (Dec 2017). 10.1007/978-3-319-70694-8_15

[17] Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (Jan 2020). 10.1007/s00145-019-09319-x

[18] Cho, J., Ha, J., Kim, S., Lee, B., Lee, J., Lee, J., Moon, D., Yoon, H.: Transciphering framework for approximate homomorphic encryption. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 640–669. Springer (2021)

[19] Cohen, A., Cousins, D.B., Genise, N., Kline, E., Polyakov, Y., RV, S.: HRA-secure homomorphic lattice-based proxy re-encryption with tight security. IACR Communications in Cryptology **2**(1) (2025). 10.62056/ab0l5wol7

[20] Cowan, M., Dangwal, D., Alaghi, A., Trippel, C., Lee, V.T., Reagen, B.: Porcupine: A synthesizing compiler for vectorized homomorphic encryption. In: Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation. pp. 375–389 (2021)

[21] Ducas, L., Micciancio, D.: FHEW: Bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 617–640. Springer, Berlin, Heidelberg (Apr 2015). 10.1007/978-3-662-46800-5_24

[22] Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), https://eprint.iacr.org/2012/144

[23] Gorantala, S., Springer, R., Purser-Haskell, S., Lam, W., Wilson, R., Ali, A., Astor, E.P., Zukerman, I., Ruth, S., Dibak, C., et al.: A general purpose transpiler for fully homomorphic encryption. arXiv preprint arXiv:2106.07893 (2021)

[24] Grassi, L., Ayala, I.M., Hovd, M.N., Øygarden, M., Raddum, H., Wang, Q.: Cryptanalysis of symmetric primitives over rings and a key recovery attack on rubato. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part III. LNCS, vol. 14083, pp. 305–339. Springer, Cham (Aug 2023). 10.1007/978-3-031-38548-3_11

[25] Guo, Q., Nabokov, D., Suvanto, E., Johansson, T.: Key recovery attacks on approximate homomorphic encryption with {Non-Worst-Case} noise flooding countermeasures. In: 33rd USENIX Security Symposium (USENIX Security 24). pp. 7447–7461 (2024)

[26] Ha, J., Kim, S., Lee, B., Lee, J., Son, M.: Rubato: Noisy ciphers for approximate homomorphic encryption. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part I. LNCS, vol. 13275, pp. 581–610. Springer, Cham (May / Jun 2022). 10.1007/978-3-031-06944-4_20

[27] Kluczniak, K., Santato, G.: On circuit private, multikey and threshold approximate homomorphic encryption. CiC **2**(1), 9 (2025). 10.62056/a69qgyl7s

[28] Li, B., Micciancio, D.: On the security of homomorphic encryption on approximate numbers. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 648–677. Springer, Cham (Oct 2021). 10.1007/978-3-030-77870-5_23

[29] Li, B., Micciancio, D., Schultz, M., Sorrell, J.: Securing approximate homomorphic encryption using differential privacy. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part I. LNCS, vol. 13507, pp. 560–589. Springer, Cham (Aug 2022). 10.1007/978-3-031-15802-5_20

[30] Liu, F., Kalam, A., Sarkar, S., Meier, W.: Algebraic attack on fhe-friendly cipher hera using multiple collisions. IACR Transactions on Symmetric Cryptology **2024**(1), 214–233 (2024)

[31] Liu, Z., Wang, Y., Fisch, B.: IND-CPA-D of relaxed functional bootstrapping: A new attack, A general fix, and A stronger model. In: Huang, C.Y., Chen, J.C., Shieh, S.P., Lie, D., Cortier, V. (eds.) ACM CCS 2025. pp. 2907–2921. ACM Press (Oct 2025). 10.1145/3719027.3765170

[32] jie Lu, W., Huang, Z., Hong, C., Ma, Y., Qu, H.: PEGASUS: Bridging polynomial and non-polynomial evaluations in homomorphic encryption. In: 2021 IEEE Symposium on Security and Privacy. pp. 1057–1073. IEEE Computer Society Press (May 2021). 10.1109/SP40001.2021.00043

[33] Niu, C., Wei, B., Huang, Z., Yang, Z., Hong, C., Wang, M., Wei, T.: Sok: Fhe-friendly symmetric ciphers and transciphering. Cryptology ePrint Archive (2025)

[34] Schneider, T., Yalame, H., Yonli, M.: Griffin: Towards mixed multi-key homomorphic encryption. Cryptology ePrint Archive (2023)

[35] Vershynin, R.: High-Dimensional Probability: An Introduction with Applications in Data Science. Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press. 10.1017/9781108231596, https://www.math.uci.edu/~rvershyn/papers/HDP-book/HDP-book.html

[36] Viand, A., Jattke, P., Haller, M., Hithnawi, A.: Heco: Automatic code optimizations for efficient fully homomorphic encryption. arXiv preprint arXiv:2202.01649 (2022)

[37] Wang, Z., He, H., Zhao, L., Li, P., Li, Z., Meng, D., Hou, R.: Chameleon: An efficient fhe scheme switching acceleration on gpus. arXiv preprint arXiv:2410.05934 (2024)