# Key Recovery Attacks on Approximate Homomorphic Encryption with Non-Worst-Case Noise Flooding Countermeasures

Qian Guo and Denis Nabokov, *Lund University;* Elias Suvanto, *ENS Lyon;* Thomas Johansson, *Lund University*

https://www.usenix.org/conference/usenixsecurity24/presentation/guo-qian

This paper is included in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

# Key Recovery Attacks on Approximate Homomorphic Encryption with Non-Worst-Case Noise Flooding Countermeasures

Qian Guo
*Lund University*

Denis Nabokov
*Lund University*

Elias Suvanto
*ENS Lyon*

Thomas Johansson
*Lund University*

## Abstract

In this paper, we present novel key-recovery attacks on Approximate Homomorphic Encryption schemes, such as CKKS, when employing noise-flooding countermeasures based on non-worst-case noise estimation. Our attacks build upon and enhance the seminal work by Li and Micciancio at EUROCRYPT 2021. We demonstrate that relying on average-case noise estimation undermines noise-flooding countermeasures, even if the secure noise bounds derived from differential privacy as published by Li et al. at CRYPTO 2022 are implemented. This study emphasizes the necessity of adopting worst-case noise estimation in Approximate Homomorphic Encryption when sharing decryption results.

We perform the proposed attacks on OpenFHE, an emerging open-source FHE library garnering increased attention. We experimentally demonstrate the ability to recover the secret key using just one shared decryption output. Furthermore, we investigate the implications of our findings for other libraries, such as IBM's HElib library, which allows experimental estimation of the noise bounds. Finally, we reveal that deterministic noise generation utilizing a pseudorandom generator fails to provide supplementary protection.

## 1 Introduction

Fully homomorphic encryption (FHE) is a cryptographic primitive that supports arbitrary computations on encrypted data without using the decryption key and without revealing information about plaintexts. This problem appeared already in the 70s [29] but a solution did not appear until 2009, when Gentry [18] constructed the first such schemes. Many homomorphic encryption schemes [6, 12, 15, 17, 19] have now appeared and most of them are based on the security of the Learning with Errors (LWE) problem or some of its variants. In 2022, Brakerski, Gentry, and Vaikuntanathan received the prestigious Gödel prize for their transformative contributions to cryptography by constructing efficient FHE schemes.

FHE is a very powerful cryptographic tool and can be a solution to many security problems, particularly for distributed or outsourced computation. A future application area is also in AI and privacy-preserving machine learning.

A very popular FHE branch is *approximate* homomorphic encryption such as CKKS [10] and its variants [7, 8]. FHE on approximate numbers, as proposed by Cheon, Kim, Kim and Song [10], has attracted much attention from the fact that it improves the efficiency of computing on encrypted data in many applications where approximate results are acceptable, like privacy-preserving machine learning. FHE based on LWE builds ciphertexts that contain noise and as operations on ciphertexts are performed, the noise is increasing. One has to control the noise and make sure it does not grow to be too large, in order to ensure correct decryption. The main contribution in CKKS was the observation that it can be tolerable for decryption to be approximate so that the plaintext from the decrypted ciphertext is only approximately the correct one. Typically, this is the case in applications where small errors can occur. The CKKS scheme then supports real-valued plaintexts, making it efficient and attractive for certain applications, allowing for some small errors in the least significant bits. CKKS has since been implemented in many well-known open-source libraries for homomorphic encryption and also been extensively examined and optimized.

The CKKS scheme as most other lattice-based FHE schemes meets the basic security notion of indistinguishability under chosen plaintext attack (IND-CPA), assuming the hardness of some well-studied problems, like the average-case hardness of the Learning With Errors (LWE) problem which in turn relates to the worst-case complexity of some computational problems on lattices. Achieving a stronger security notion – indistinguishability under chosen ciphertext attack (IND-CCA), which presumes an active adversary capable of tampering with (or injecting) arbitrary ciphertexts – is challenging, due to the inherent nature of FHE operations.

Li and Micciancio [23] have recently shown that the traditional formulation of IND-CPA security does not adequately capture the security of approximate encryption against passive attacks. Specifically, in the context of a homomorphic encryption scheme, a passive adversary has the capability to select or know the homomorphic function that is being executed. Such an adversary can also observe the outcome of the decryption process. Despite being able to eavesdrop, this adversary remains categorized as passive, as it is not permitted to manipulate, inject, or modify ciphertexts, nor alter the final computational result. Li and Micciancio have shown that the CKKS scheme is vulnerable to an efficient key recovery attack under these conditions.

They also proposed a new, enhanced formulation of IND-CPA security that they called IND-CPA with decryption oracles (IND-CPA$^{\mathsf{D}}$). This notion captures better the threats of a passive attacker. It is proven to be identical to IND-CPA security for encryption schemes with exact decryption, but the situation is different when allowing approximate decryption. The primary distinction lies in the utility of observing decryption outcomes. In encryption schemes with exact decryption, an adversary gains no additional information from observing the decryption result. This is because the adversary, who already knows the original message and evaluation function, can calculate the final outcome on its own. Conversely, for approximate encryption schemes, observing the decryption output could yield invaluable information not readily computable by the adversary. Such information may be exploited by a passive adversary for effective key recovery.

A relevant security notion is KR$^{\mathsf{D}}$ security against key recovery attacks, a notion that, while implied by IND-CPA$^{\mathsf{D}}$ security, is significantly weaker. This weakness arises because key recovery attacks constitute a more potent form of attack compared to distinguishing attacks.

Li and Micciancio also suggested some ideas of countermeasures to avoid the proposed attack, and the open-source libraries implementing CKKS include different but similar countermeasures to thwart the attack in [23]. Subsequently, Li, Micciancio, Schultz, and Sorrell further examined the use of differential privacy to assure IND-CPA$^{\mathsf{D}}$ security in [24]. Their method necessitates the inclusion of a considerable amount of additional noise.

FHE schemes, and CKKS-like schemes in particular, involve different parameters and a major task for the designer of the scheme is the choice of parameters that should give the best trade-off between efficiency and security. Both these aspects relate to the connected noise growth in the scheme. For approximate FHE, encoding noise and encryption noise should be considered together. Also, we have to keep track of the level of ciphertexts as in traditional FHE schemes as well as a so-called scaling factor. CKKS-like schemes consider the precision of the scheme and one has to track how it changes through homomorphic operations and its influence on the encrypted data. The scale parameter is typically a way for the user to control the desired precision.

The CKKS paper provided an open-source implementation in the "Homomorphic Encryption for Arithmetic on Approximate Numbers", Heaan [1], other implementations of the scheme have been included in essentially all mainstream libraries for secure computation on encrypted data, like Microsoft's "Simple Encrypted Arithmetic Library" SEAL [26], IBM's "Homomorphic Encryption" library HElib [21], the lattice cryptography libraries PALISADE [2] and Lattigo [16]. Recently, the PALISADE authors along with a subset of authors from the other libraries released a new library in July 2022, under the name OpenFHE [4]. OpenFHE provides implementations of all main FHE schemes.

These libraries are also used as a part of other implemented tools and applications. A well-known example is Intel's nGraph-HE compiler [5] for secure machine learning applications. Other implemented applications include the encrypted computation of logistic regression [20], security-preserving support vector machine [27], homomorphic evaluation of neural networks and tensor programs [14], and clustering over encrypted data [11].

A recent trend in the implementation of CKKS-like schemes involves using a non-worst-case noise estimation, such as an empirical or average-case noise estimation, as seen in sources like [4, 13, 21]. In this context, a worst-case noise bound refers to a bound that tracks the maximum possible noise throughout each step of homomorphic evaluation. Conversely, empirical noise estimation involves setting the noise limit based on empirical tests, while average-case noise estimation models the noise as a Gaussian distribution, keeping tracks of the variance during each homomorphic encryption operation. Opting for a non-worst-case approach can enhance performance, since the non-worst-case noise bound can be significantly smaller than the worst-case noise bound, thus substantially improving the scheme's efficiency. This is particularly appealing when considering the efficiency loss incurred while protecting against the Li-Micciancio attack.

To our knowledge, OpenFHE is the first major open-source library to claim the implementation of the differential privacy countermeasure as proposed in [24], yet they suggest the use of a non-worst-case noise estimation. In this paper, we explore the potential impact of such noise estimations on security, when the decryption results need to be shared.

## 1.1 Contributions

The main contributions of this work can be summarized as follows. Firstly, using non-worst-case estimation methods, such as relying on the central limit theorem (CLT) for average-case noise estimation [13] or experimentally determining the noise bound, can introduce significant security vulnerabilities. For instance, even when adopting the secure noise bounds suggested in [24] to achieve IND-CPA$^{\mathsf{D}}$ security through differential privacy, KR$^{\mathsf{D}}$ security may not be ensured if the noise size is estimated in a non-worst-case manner. The novel observation is that such non-worst-case estimation assumes that the message distribution or the computation gate should follow some average-case or predefined behaviour; however, in the IND-CPA$^{\mathsf{D}}$ attacker model, the adversary has the power to control the message distribution, causing it to deviate significantly from the average case. The adversary also has the power to select the functions and gates among the possible valid options.

We implement a key-recovery attack on the recent OpenFHE implementation, requiring only a single shared decryption output. The OpenFHE library, a successor to PALISADE, is chosen as the primary target implementation since it incorporates the noise-flooding countermeasure with the new secure noise bounds [24] derived from differential privacy[1]. In contrast, Microsoft SEAL disallows the sharing of decryption results, while open-source HElib and HEAAN have not implemented countermeasures based on the latest CRYPTO 2022 research [24]. We are capable of attacking OpenFHE as its noise estimation method aligns with the average-case noise estimation proposed in [13].

The experimental noise estimation employed in HElib necessitates that the server performing homomorphic evaluation adheres to the message/ciphertext distribution and the evaluation gates assumed by the user. However, these requirements contradict the adversary model of IND-CPA$^{D}$ security. Our analysis indicates that the flooding countermeasure with such noise estimation is also insecure.

Finally, as an additional finding, we show that the protection of utilizing a pseudorandom generator to produce deterministic noise with the ciphertext serving as an input seed does not improve security. Such a countermeasure has been suggested in [9,23] and implemented in HElib. We have carefully designed a mechanism to bypass this security measure. The code for all of our experiments is public[2].

### 1.2 Organizations

The remainder of this paper adheres to the following organization. Section 2 introduces the background on Approximate Homomorphic Encryption schemes. Section 3 presents the attack targeting implementations using non-worst-case noise estimation. Section 4 describes the attack method for deterministic noise estimation for "fresh ciphertexts". Lastly, Section 5 offers concluding insights and additional discussions.

## 2 Preliminaries

In this section, we provide essential background information on Approximate Homomorphic Encryption schemes and describe their security notions.

**Notation.** Let $\mathbb{Z},\mathbb{Q},\mathbb{R},\mathbb{C}$ represent the sets of integers, rationals, reals, and complex numbers, respectively. Bold characters, such as $\mathbf{a} = (a_1,\ldots,a_n)$, denote vectors. The dot product of two vectors is written as $\langle \mathbf{a},\mathbf{b}\rangle$. For a probability distribution $\mathcal{D}$, we use $x \leftarrow \mathcal{D}$ to indicate that $x$ is sampled according to the distribution $\mathcal{D}$. We denote the additional noise of countermeasures with $\varepsilon$, which, though not necessarily small, can be considered as random computational noise. Let $\lfloor \cdot \rceil$ denote the operation of rounding to the nearest integer. The security parameter is represented by $\kappa$. A function $f$ is deemed negligible in $\kappa$ if $f(\kappa) = \kappa^{-\omega(1)}$.

[1]https://github.com/openfheorg/openfhe-development/blob/main/src/pke/examples/CKKS_NOISE_FLOODING.md.

[2]https://github.com/d-nabokov/KRAonCKKS

### 2.1 Lattices and the Ring-LWE problem

A lattice is a discrete subgroup of $\mathbb{R}^n$. Let $n,d$ be two positive integers, and let $\mathbf{b}_1,\ldots,\mathbf{b}_d$ be linearly independent vectors in $\mathbb{R}^n$. The lattice they generate is $\mathcal{L}(\mathbf{b}_1,\ldots,\mathbf{b}_d) = \sum_{i=1}^{d} \mathbb{Z}\mathbf{b}_i$. If $d = n$, the lattice is considered full-rank.

Let $N = 2^k$ be a power of 2. Let $\psi(N) = X^N + 1$ be a power-of-2 cyclotomic polynomial. Let $R = \mathbb{Z}[X]/\psi(N)$. For $b \in R$, we denote $\|b\|_2$ the Euclidean norm of the vector of its coefficients. Let $R_q = \mathbb{Z}_q[X]/\psi(N) = R/qR$.

We define $\mathcal{N}(\mu,\sigma)$ as the discrete Gaussian distribution with mean $\mu$ and standard deviation $\sigma$ – that is, a probability distribution supported on $\mathbb{Z}$ with a probability mass function $p(x) \propto \exp(-(x-\mu)^2/(2\sigma^2))$. Such a distribution can be approximated by its continuous counterpart. Specifically, if $x \leftarrow \mathcal{N}(\mu_1,\sigma_1)$ and $y \leftarrow \mathcal{N}(\mu_2,\sigma_2)$, respectively, the sum $x+y$ is regarded as a draw from $\mathcal{N}(\mu_1+\mu_2, \sqrt{\sigma_1^2+\sigma_2^2})$. We can make the distribution over $\mathbb{Z}_q$ by reducing the output mod $q$.

The best-known FHE schemes are built upon the Learning with Errors (LWE) problem [28]. In the CKKS scenario, we focus more on its ring version, known as the Ring-LWE problem [25]. The Ring-LWE distribution is obtained as follows.

**Definition 1** *Let $k \geq 1, q \geq 2, \alpha \in \mathbb{R}, s \in R_q$. The Ring-LWE distribution is obtained as follows:*

$$a \leftarrow \mathcal{U}(R_q);$$

$$e \leftarrow R \quad \textit{with coefficients } \leftarrow \mathcal{N}(0, \frac{\alpha q}{\sqrt{2\pi}});$$

*The oracle subsequently returns $(a, a\cdot s + e) \in R_q \times R_q$.*

### 2.2 Approximate Homomorphic Encryption

In their work [10], Cheon et al. introduced the first approximate homomorphic encryption scheme, commonly referred to as CKKS, named after its creators. Here, we first present the definition of Homomorphic Encryption. Gentry's bootstrapping blueprint [18] allows for the construction of a Fully Homomorphic Encryption (FHE) scheme from an HE scheme.

**Definition 2 (Homomorphic Encryption (HE))** *A (public-key) HE scheme comprises four polynomial time algorithms:*

- *A (randomized) key generation algorithm* $\mathsf{KeyGen} : 1^\kappa \rightarrow \mathcal{PK}^2 \times \mathcal{SK}$ *that generates the public key, evaluation key, and secret key.*
- *A (randomized) encryption algorithm* $\mathsf{Enc} : \mathcal{PK} \times \mathcal{M} \rightarrow \mathcal{C}$.
- *A (deterministic) decryption algorithm* $\mathsf{Dec} : \mathcal{SK} \times \mathcal{C} \rightarrow \mathcal{M}$.
- *A (deterministic) homomorphic evaluation function* $\mathsf{Eval}\ \forall k, \mathcal{PK} \times (\mathcal{M}^k \rightarrow \mathcal{M}) \times \mathcal{C}^k \rightarrow \mathcal{C}$.

In the given definition, $\mathcal{PK}$ signifies the collection of public keys, inclusive of the evaluation keys, while $\mathcal{SK}$ denotes the set of secret keys. Additionally, $\mathcal{M}$ represents the message space, and $\mathcal{C}$ stands for the ciphertext space. The function provided to the evaluation method is a circuit comprising multiple gates.

In standard HE schemes, the correctness of the scheme is defined as

**Definition 3 (HE Correctness)** *For an arbitrary circuit $\mathfrak{C}$ the following holds*

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Eval}_{\mathsf{ek}}(\mathfrak{C}, (c_1, \dots, c_k))) = \mathfrak{C}(m_1, \dots, m_k),$$

*where* $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}) = \mathsf{KeyGen}(1^\kappa)$ *and* $c_i = \mathsf{Enc}_{\mathsf{pk}}(m_i)$ *for all* $i \leq k$.

**Approximate HE schemes** Approximate HE schemes are a special type of HE schemes where the notion of correctness does not hold. They allow approximate arithmetic on encrypted data, considering errors of homomorphic operations as part of the computational errors in the approximate computation. As the first approximate HE proposal, CKKS supports two approximate homomorphic operations:

- $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Add}(\mathsf{Enc}_{\mathsf{pk}}(m_1), \mathsf{Enc}_{\mathsf{pk}}(m_2))) \simeq m_1 + m_2$,
- $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Mult}(\mathsf{Enc}_{\mathsf{pk}}(m_1), \mathsf{Enc}_{\mathsf{pk}}(m_2))) \simeq m_1 * m_2$,

with a tracker on the size of the noise. Next, we will delve into the specifics of the CKKS scheme, beginning with its method for message encoding.

**The CKKS message encoding.** In order to perform pointwise addition and multiplication of vectors in the SIMD manner, the CKKS scheme designs the following encoding method. Let $\omega$ be the $2N$-th root of unity. Let $\phi : R \rightarrow \mathbb{C}$ be the mapping from $a(X) \in R$ to $\phi(a) = \hat{\mathbf{a}} = (a(\omega^{4j+1}))_{j=0}^{N/2-1} \in \mathbb{C}^{N/2}$. The CKKS scheme chooses a positive real $\Delta$ called scaling factor and defines the encoding and decoding functions as

- $\mathsf{Encode}(\mathbf{z} \in \mathbb{C}^{N/2}; \Delta) = \lfloor \Delta \cdot \phi^{-1}(\mathbf{z}) \rceil$,
- $\mathsf{Decode}(a \in R; \Delta) = \phi(\Delta^{-1} \cdot a) \in \mathbb{C}^{N/2}$,

where $\phi^{-1}$ is the inverse function of $\phi$.

**The CKKS scheme [10].** The CKKS scheme supports the encryption and decryption of floating-point inputs using the message encoding interface. We now consider the plaintext space as $R_q$ and the ciphertext space as $R_q^2$. We have a ciphertext module $q$ and a discrete Gaussian error distribution $\chi$, where $\chi = \mathcal{N}(0, \sigma)$. We bypass many intricate details, focusing on the primary procedures and attributes pertinent to this study. The following description is from [24]. For a more comprehensive understanding, we refer the reader to [10].

- $\mathsf{CKKS.KeyGen}(1^\kappa)$: Let $w = w(\kappa)$ and $p = p(\kappa, q)$. We sample $s$ from the set $s \in \{-1, 0, 1\}^N$ with $w$ non-zero entries in $s$ and designate $\mathsf{sk}$ as $(1, -s)$. Alternatively, $s$ can be sampled from a uniform distribution over $\{-1, 0, 1\}^N$. We sample $a \leftarrow R_q$, $e \leftarrow \chi$ and take $\mathsf{pk}= (a, b)$ with $b = a \cdot s + e$. We sample $a' \leftarrow R_{pq}$, $e' \leftarrow \chi$ and take $\mathsf{ek}= (a', b')$ with $b' = a' \cdot s + e + p \cdot s^2 \mod pq$. Return ($\mathsf{sk}$, $\mathsf{pk}$, $\mathsf{ek}$).
- $\mathsf{CKKS.Enc}_{\mathsf{pk}}(m)$: Let $\xi$ represent the distribution over $\{-1, 0, 1\}^N$, where each coefficient is independently sampled as 1 or $-1$ with a probability of $1/4$, and 0 with a probability of $1/2$. Then, we sample $r \leftarrow \xi$, $e_0, e_1 \leftarrow \chi$, and return $r \cdot \mathsf{pk} + (m + e_0, e_1) \mod q$.
- $\mathsf{CKKS.Add}(c_0, c_1)$: Return $c_0 + c_1 \mod q$.
- $\mathsf{CKKS.Mult}_{\mathsf{ek}}(c_0, c_1)$: Assume $c_0 = (a_0, b_0)$ and $c_1 = (a_1, b_1)$. Return $(a_2, b_2)$, where $(a_2, b_2) = (a_0 b_1 + a_1 b_0, b_0 b_1) + \lfloor p^{-1} \cdot a_0 a_1 \cdot \mathsf{ek} \rceil \mod q$.
- $\mathsf{CKKS.Dec}_{\mathsf{sk}}(c)$: For $c = (a, b) \in R_q^2$, return $b - a \cdot s \mod q$.

Analyzing the addition operation in the worst-case scenario can be done straightforwardly. Suppose $c_0$ and $c_1$ are two ciphertexts with positive error bounds $errb_0$ and $errb_1$, respectively. The ciphertext resulting from the $\mathsf{Add}$ operation on $c_0$ and $c_1$ would have an error bounded to $errb_0 + errb_1$.

## 2.3 Security notions

The standard notion of security for FHE is IND-CPA (Indistinguishability - Chosen Plaintext Attack) security defined as follows.

**Definition 4** (IND-CPA **security**) *We say that a FHE scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *is* IND-CPA ***secure*** *if each efficient adversary $\mathcal{A}$ has negligible advantage* $\mathsf{Adv}_{\mathcal{A}}(\kappa)$*, where the advantage is defined as*

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = \left| \Pr\left[\mathcal{A} \xrightarrow{\mathsf{Expr}_0} 1\right] - \Pr\left[\mathcal{A} \xrightarrow{\mathsf{Expr}_1} 1\right] \right|.$$

*The experiments are as follows :*

$$\begin{array}{llcl}
\mathsf{Expr}_b: & \mathcal{C} & & \mathcal{A} \\
 & (\mathsf{sk},\mathsf{pk},\mathsf{ek}) \leftarrow \mathsf{KeyGen} & \xrightarrow{\mathsf{pk}} & \\
 & c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b) & \xleftarrow{m_0,m_1} & \\
 & & \xrightarrow{c} & \textit{Outputs a bit } b
\end{array}$$

An additional security notion, IND-CPA$^D$, proves vital when the decryption outputs require sharing, as pointed out in [23]. For non-approximate HE, the IND-CPA$^D$ notion is proven to be equivalent to the IND-CPA notion in [23]. However, for approximate HE, it represents a much stronger security notion. We present a formal definition as follows.

**Definition 5** (IND-CPA$^{\mathsf{D}}$ **security [23]**) *Given a public-key homomorphic (possibly approximate) encryption scheme $\mathcal{E}$, where $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$, with plaintext space $\mathcal{M}$ and ciphertext space $\mathcal{C}$, we define an experiment $\mathsf{Expr}_b^{\mathsf{indcpa}^D}[\mathcal{A}]$, parameterized by a bit $b \in \{0,1\}$ and involving an efficient adversary $\mathcal{A}$ that is given access to the following oracles, sharing a common state $S \in (\mathcal{M} \times \mathcal{M} \times \mathcal{C})^*$ consisting of a sequence of message-message-ciphertext triples:*

- *An encryption oracle $\mathsf{E}_{\mathsf{pk}}(m_0, m_1)$ that, given a pair of plaintext messages $m_0, m_1$, computes $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_b)$, extends the state*

  $$S := [S; (m_0, m_1, c)],$$

  *with one more triplet, and returns the ciphertext $c$ to the adversary.*

- *An evaluation oracle $\mathsf{H}_{\mathsf{ek}}(g, J)$ that, given a function $g : \mathcal{M}^k \to \mathcal{M}$ and a sequence of indices $J = (j_1, \ldots, j_k) \in \{1, \ldots, |S|\}^k$, computes the ciphertext $c \leftarrow \mathsf{Eval}_{\mathsf{ek}}(g, S[j_1].c, \ldots, S[j_k].c)$, extends the state*

  $$S := [S; (g(S[j_1].m_0, \ldots, S[j_k].m_0), \\ g(S[j_1].m_1, \ldots, S[j_k].m_1), c)]$$

  *with one more triplet, and returns the ciphertext $c$ to the adversary. Here and below $|S|$ denotes the number of triplets in the sequence $S$, and $S[j].m_0, S[j].m_1, S[j].c$ denote the three components of the $j$-th element of $S$.*

- *A decryption oracle $\mathsf{D}_{\mathsf{sk}}(j)$ that, given an index $j \le |S|$, checks whether $S[j].m_0 = S[j].m_1$, and, if so, returns $\mathsf{Dec}_{\mathsf{sk}}(S[j].c)$ to the adversary. (If the check fails, a special error symbol $\perp$ is returned.)*

*The experiment is defined as*

$$\begin{aligned}
\mathsf{Expr}_b^{\mathsf{indcpa}^D}[\mathcal{A}](1^\kappa):\ & (\mathsf{sk},\mathsf{pk},\mathsf{ek}) \leftarrow \mathsf{KeyGen}(1^\kappa) \\
& S := [] \\
& b' \leftarrow \mathcal{A}^{\mathsf{E}_{\mathsf{pk}},\mathsf{H}_{\mathsf{ek}},\mathsf{D}_{\mathsf{sk}}}(1^\kappa, \mathsf{pk}, \mathsf{ek}) \\
& \mathsf{return}(\mathsf{b}')
\end{aligned}$$

*The advantage of adversary $\mathcal{A}$ against the* IND-CPA$^D$ *security of the scheme is $\mathsf{Adv}_{\mathsf{indcpa}^D}[\mathcal{A}](\kappa)$ defined as*

$$\left| \mathbf{Pr}\left[\mathsf{Expr}_0^{\mathsf{indcpa}^D}[\mathcal{A}(1^\kappa)] = 1\right] - \mathbf{Pr}\left[\mathsf{Expr}_1^{\mathsf{indcpa}^D}[\mathcal{A}(1^\kappa)] = 1\right] \right|,$$

*where the probability is over the randomness of $\mathcal{A}$ and the experiment. The scheme $\mathcal{E}$ is* IND-CPA$^D$*-secure if for any efficient (probabilistic polynomial time) $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathsf{indcpa}^D}[\mathcal{A}](\kappa)$ is negligible in $\kappa$.*

Another relevant security notion is the KR$^{\mathsf{D}}$ security, implied by the IND-CPA$^{\mathsf{D}}$ security. Although KR$^{\mathsf{D}}$ security is considered much weaker than the IND-CPA$^{\mathsf{D}}$ security, a key-recovery attack is predominantly employed to demonstrate the vulnerability of the analyzed schemes. This methodology aligns with the research presented in this paper. We present the formal definition of KR$^{\mathsf{D}}$ security in Definition 6.

**Definition 6** (KR$^{\mathsf{D}}$ **security [24]**) *Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a public-key homomorphic (possibly approximate) encryption scheme with plaintext space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. We define a game $\mathsf{Expr}^{\mathsf{KR}^{\mathsf{D}}}[\mathcal{A}]$, parameterized by an adversary $\mathcal{A}$ that is given access to the (stateful) oracles $\mathsf{E}'_{\mathsf{pk}}$, $\mathsf{H}_{\mathsf{ek}}$, $\mathsf{D}_{\mathsf{sk}}$, where $\mathsf{E}'_{\mathsf{pk}}(m) := \mathsf{E}_{\mathsf{pk}}(m, m)$ with oracles $\mathsf{E}_{\mathsf{pk}}$, $\mathsf{H}_{\mathsf{ek}}$, $\mathsf{D}_{\mathsf{sk}}$ defined as in Definition 5.*

*The game is defined as*

$$\begin{aligned}
& \mathsf{Expr}^{\mathsf{KR}^{\mathsf{D}}}[\mathcal{A}](1^\kappa): \\
& \qquad (\mathsf{sk},\mathsf{pk},\mathsf{ek}) \leftarrow \mathsf{KeyGen}(1^\kappa) \\
& \qquad S := [] \\
& \qquad \mathsf{sk}' \leftarrow \mathcal{A}^{\mathsf{E}'_{\mathsf{pk}},\mathsf{H}_{\mathsf{ek}},\mathsf{D}_{\mathsf{sk}}}(1^\kappa, \mathsf{pk}, \mathsf{ek}) \\
& \qquad \mathsf{return}(\mathsf{sk} == \mathsf{sk}')
\end{aligned}$$

*The scheme $\mathcal{E}$ is said to have $k$ bits of* KR$^{\mathsf{D}}$*-security if, for any adversary $\mathcal{A}$,*

$$k \le \log_2 \frac{T(\mathcal{A})}{\mathsf{Adv}_{\mathcal{A}}},$$

*where $T(\mathcal{A})$ is the running time of $\mathcal{A}$.*

In [24], the application of differential privacy technique results in a finer-grained definition of bit-security, parameterized by both a computational parameter $c$ and a statistical parameter $\nu$. We define this security notion as follows.

**Definition 7** (($c,\nu$)**-bits of security [24]**) *Let $\Pi$ be a cryptographic primitive, and $\mathcal{G}$ be an indistinguishability game. Let $\mathsf{Adv}_{\mathcal{A}}$ be the advantage of an adversary $\mathcal{A}$ in breaking the security of $\Pi$ in the $\mathcal{G}$ game. The scheme $\Pi$ has $(c,\nu)$-bits of $\mathcal{G}$-security if, for any adversary $\mathcal{A}$, either*

$$\log_2 \frac{T(\mathcal{A})}{\mathsf{Adv}_{\mathcal{A}}} \ge c, \quad or, \quad \log_2 \frac{1}{\mathsf{Adv}_{\mathcal{A}}} \ge \nu. \tag{1}$$

## 3 New key-recovery attacks

In this section, we improve upon key-recovery attacks initially presented by Li and Micciancio [23], when non-worst-case noise estimation is employed. We begin with the theoretical framework, then transition to the security implications for the OpenFHE and HElib homomorphic encryption libraries.

### 3.1 Theoretical framework

**Attack model.** We adopt the adversary model defined in [23], as also illustrated in Definition 6 and Figure 1. In this model, the adversary gains access to three stateful oracles $\mathsf{E}'_{\mathsf{pk}}$, $\mathsf{H}_{\mathsf{ek}}$, and $\mathsf{D}_{\mathsf{sk}}$, with $\mathsf{E}'_{\mathsf{pk}}(m) := \mathsf{E}_{\mathsf{pk}}(m,m)$. These oracles are defined in Definition 5. The adversary aims to recover the secret key $\mathsf{sk}$ and is capable of querying the three oracles, selecting messages and evaluation functions as inputs. Notably, this adversary remains passive; unlike an active adversary who can manipulate arbitrary ciphertexts, this model restricts the adversary to decryption queries on valid ciphertexts produced through legitimate operations, such as encryption and homomorphic computation via public interfaces or APIs provided by HE libraries.

**Real-world scenarios.** In practical applications, it is highly likely that decrypted data will be shared among parties who lack access to the secret key, making IND-CPA$^{\mathsf{D}}$ security indispensable. This sharing could facilitate significant applications like two-party computation for Privacy-Preserving Machine Learning as a Service (MLaaS), wherein the server owns a model it wishes to provision as a service, while the client seeks to obtain a homomorphically computed inference on its confidential data. The ecosystem might comprise a user holding the secret key and other participants lacking it, who function as passive adversaries. Service providers contribute by supplying APIs and computational resources.

An interesting case study involves a commercial application that collects encrypted data from the user and leverages a cloud service's API for homomorphic computations. We posit that the application can only invoke the API using ciphertexts authenticated by the user, thereby precluding the tampering or injection of fraudulent ciphertexts. The API invocation resembles the usage of the evaluation oracle $\mathsf{H}_{\mathsf{ek}}$ as outlined in Definition 5. Should the user disclose the decrypted outcome, our novel attack strategy remains viable, even if non-worst-case noise-flooding countermeasures have been deployed.

**The Li-Micciancio attack.** Similar to [23], we use a simplified CKKS scheme without the encoding and decoding procedures as an example to explain the attack idea. The secret key is $\mathsf{sk}{=}(1,-s)$, where $s$ is a power-of-two cyclotomic ring element in $R_q$. The only unknown part of the secret key is $s$, so the recovery of $\mathsf{sk}$ is equivalent to recovering $s$. The adversary encrypts $m$ as $\mathsf{Enc}(m) = (a,b) \in R_q^2$, where $a \in R_q$ is selected at random and $b = a \cdot s + e + m$ for a polynomial $e \in R_q$ where the coefficients of $e$ are a small perturbation from the distribution $\mathcal{N}(0,\sigma_1)$. The decryption function computes $\mathsf{Dec}_s((a,b)) = b - a \cdot s = m + e$. In the Li-Micciancio attack, the adversary $\mathcal{A}$ selects the message $m = 0$ and requests the computation of the identity function, effectively equivalent to performing no operation. The adversary then proceeds to request an approximate decryption and carries out subsequent computations of

$$b' = b - \mathsf{Dec}_s((a,b)) = a \cdot s.$$

The secret key $s$ can then be recovered by computing

$$s = a^{-1} \cdot b',$$

if $a$ is an invertible element in $R_q$. Otherwise, the adversary requires for more decryption queries.

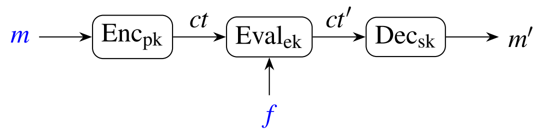

$m$
$\mathrm{Enc}_{\mathrm{pk}}$
$ct$
$\mathrm{Eval}_{\mathrm{ek}}$
$ct'$
$\mathrm{Dec}_{\mathrm{sk}}$
$m'$
$f$

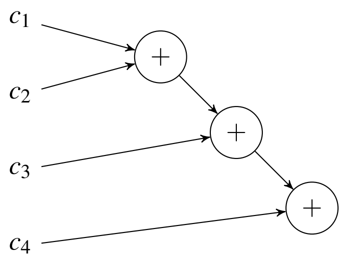
Figure 1: Figure illustrates a passive attacker scenario based on Li and Micciancio's model [23]. In this context, the attacker may choose/know the plaintext $m$ and the homomorphic function $f$. The attacker can eavesdrop on the ciphertexts $ct$, $ct'$ and decryption result $m'$, but is not allowed to tamper with (or inject) ciphertexts or alter the computation's final outcome.

**Noise-flooding countermeasures.** Li and Micciancio have proposed a countermeasure from the noise-flooding technique, for protecting against their attack in [23]. The main idea is to introduce an additional noise in the decryption function. Thus, the decryption function of CKKS, named decryption for share, can be defined as

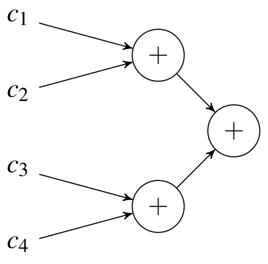
$$\mathsf{Dec}_s^{\mathsf{D}}((a,b)) = \mathsf{Dec}_s((a,b)) + e_{\mathsf{new}},$$

where $e_{\mathsf{new},i} \leftarrow \mathcal{N}(0,\sigma_2)$ and $e_{\mathsf{new},i}$ is the coefficient of $e_{\mathsf{new}}$. When the adversary $\mathcal{A}$ computes $b' = b - \mathsf{Dec}_s^{\mathsf{D}}((a,b))$, the output is $a \cdot s + e_{\mathsf{new}}$; if the noise is sufficiently large, then the new ring-LWE instance $(a, a \cdot s + e_{\mathsf{new}})$ is hard to solve.

**Achieving IND-CPA$^{\mathsf{D}}$ through differential privacy [24].** The general idea is to add a noise with sufficiently large variance. The newly proposed schemes, referred to as $\mathsf{S\text{-}CKKS}_\sigma$, are detailed in [24]. Initially, they establish a noise bound, $\mathsf{ct.t}$, designed to accommodate the worst-case noise growth, following which they add a component-wise noise, $\mathcal{N}(0, \sigma \cdot \mathsf{ct.t})$, in the noise-flooding countermeasure. In [24], the following lemma is proved.



(a) Iterative addition



(b) Tree-based addition

Figure 2: Iterative and tree-based addition for $g_t(\mathbf{c})$ with $t = 4$

**Lemma 1 ( [24])** *Assume the number of decryption oracle calls is bounded to $q$ and $\sigma = \sqrt{24qn}2^{\nu/2}$. If* CKKS *is* $(c + \log_2 24)$*-bit* IND-CPA*-secure, then* $\mathsf{S\text{-}CKKS}_\sigma$ *is* $(c, \nu)$*-bit* IND-CPA$^{\mathsf{D}}$*-secure.*

On the other hand, for recent open-source implementations (e.g., in [4]), non-worst-case noise estimation approaches are preferred to enhance efficiency.

**New attack scenario.** We describe an attack scenario when the standard deviation $\sigma_2$ of the added Gaussian noise in the noise-flooding countermeasure is a function that grows much slower than the estimated noise in the worst-case noise analysis.

In alignment with the IND-CPA$^{\mathsf{D}}$ model outlined in Definition 5, the adversary has the capability to execute computations on valid ciphertexts using a chosen function. Specifically, the adversary employs the evaluation oracle $\mathsf{H}_{\mathsf{ek}}$ for homomorphic calculations. Our primary focus is on the evaluation function $g_t(\mathbf{c})$, which performs homomorphic addition of $t$ ciphertexts $\mathbf{c} = (c_1, c_2, \ldots, c_t)$. This function, illustrated in Figure 2, can be executed through either an iterative addition over $t-1$ layers or a tree-based addition spanning $\lceil \log_2(t) \rceil$ layers. Depending on the approach, the gate latency stands at either $t-1$ or $\lceil \log_2(t) \rceil$.

In this new attack scenario, we stick with the case where $m = 0$, first generate a ciphertext $c_0 = (a, b)$, and then homomorphically evaluate the addition $g_t(\mathbf{c})$, where $\mathbf{c}$ is a vector repeating the ciphertext $c_0$ for $t$ times. The noise, therefore, is grown from $e$ to $t \cdot e$. The new decryption function $\mathsf{Dec}_s^{\mathsf{D}}(\cdot)$ gives us the value of

$$e_{\mathsf{total}} = t \cdot e + e_{\mathsf{new}}.$$

We need the following lemma to characterize the distribution of conditional probability $(e_i | e_{\mathsf{total},i} = l)$, where $e_i$ and $e_{\mathsf{total},i}$ are the $i$-th coefficient of $e$ and $e_{\mathsf{total}}$, respectively.

**Lemma 2 ( [22])** *For two positive numbers $\sigma_a$ and $\sigma_b$, let $X$ and $Y$ be variables of distributions $\mathcal{N}(0, \sigma_a)$ and $\mathcal{N}(0, \sigma_b)$, respectively. Let $\sigma = \sqrt{\sigma_a^2 + \sigma_b^2}$, then $(X | X + Y = l)$ follows the distribution $\mathcal{N}(l \cdot \frac{\sigma_a^2}{\sigma^2}, \frac{\sigma_a \sigma_b}{\sigma})$.*

In the attack, we can obtain the value of $e_{\mathsf{total}}$ from the new decryption function. Conditioned on this observation, the distribution of the noise $e$ needs to be updated. Since $t \cdot e_i$ is sampled from $\mathcal{N}(0, t \cdot \sigma_1)$ and $e_{\mathsf{new},i}$ is sampled from $\mathcal{N}(0, \sigma_2)$, from Lemma 2, we know $(t \cdot e_i | e_{\mathsf{total},i} = l)$ follows the distribution $\mathcal{N}(l \cdot \frac{t^2\sigma_1^2}{\sigma_2^2 + t^2\sigma_1^2}, \frac{t \cdot \sigma_1 \cdot \sigma_2}{\sqrt{t^2 \cdot \sigma_1^2 + \sigma_2^2}})$[3] and, hence, $(e_i | e_{\mathsf{total},i} = l)$ follows the distribution $\mathcal{N}(l \cdot \frac{t\sigma_1^2}{\sigma_2^2 + t^2\sigma_1^2}, \frac{\sigma_1 \cdot \sigma_2}{\sqrt{t^2 \cdot \sigma_1^2 + \sigma_2^2}})$. Thus, for each value of $e_{\mathsf{total},i}$, we compute

$$b'_i = b_i - e_{\mathsf{total},i} \cdot \frac{t \cdot \sigma_1^2}{\sigma_2^2 + t^2\sigma_1^2}. \tag{2}$$

The new polynomial $b'$ with the $i$-th coefficient $b'_i$ satisfies

$$b' = a \cdot s + e', \tag{3}$$

where the coefficients of $e'$ follow the distribution of $\mathcal{N}(0, \frac{\sigma_1 \cdot \sigma_2}{\sqrt{t^2 \cdot \sigma_1^2 + \sigma_2^2}})$, we use $\sigma_{\mathsf{attack}}$ to denote the standard deviation of it.

**Intuition.** We next show applications where the standard deviation of the new noise variable can be very small, making the new Ring-LWE problem easy to solve. To simplify the analysis, we let $t$ be arbitrarily large. This assumption permits us to focus on the leading terms and consider the asymptotic behavior. In the first scenario, where $\sigma_2 = \sqrt{t}\sigma_1$, the standard deviation is of the order $\Theta(t^{-1/2})$. Conversely, in the second scenario with $\sigma_2$ as a constant, the standard deviation is $\Theta(t^{-1})$. In either case, it is feasible to choose a sufficiently large $t$ to minimize the noise to negligible levels.

It is important to note that this assumption serves merely as a simplifying element for illustrative purposes. We are constrained by a noise budget and a target level of precision. In the attack scenarios explored throughout the paper, we begin

[3]In [22], the authors assume a continuous Gaussian distribution. In contrast, our approach in Lemma 2 utilizes discrete Gaussians as approximations for their continuous counterparts. Empirical tests conducted with OpenFHE's discrete Gaussian distributions, as depicted in Figure 3, affirm the precision of our approximation methodology. For an in-depth discussion, refer to Section 3.2.1.

operations using a fresh ciphertext with a small noise level. Consequently, the $t$ values required to initiate the attack do not exceed the permissible noise thresholds in the associated libraries.

**Efficient implementation of $g_t(\mathbf{c})$ for the adversary using c as a $t$-fold repetition of $c_0$.** Let $t_{\lambda_0-1}\dots t_1 t_0$ be the binary representation of $t$, where $t_0$ is the least significant bit. In the tree-based addition variant of $g_t(\mathbf{c})$, where $\mathbf{c}$ is a vector composed of $t$ repetitions of the ciphertext $c_0$, each addition operation within the same layer receives and produces identical input and output. This condition leads to the transformation of $g_t(\mathbf{c})$ into an alternate function, denoted as $g'_t(\mathbf{c})$, which effectively performs the following operations (which is similar to exponentiation by squaring[4]):

- Initialize: we assume that the most significant bit is set, i.e. $t_{\lambda_0-1} = 1$, let $c_1 \leftarrow c_0$.
- For $i$ in the range $\{1,\dots,\lambda_0 - 1\}$ iteratively compute $c_{i+1} \leftarrow \begin{cases} c_i + c_i & \text{if } t_{\lambda_0-1-i} = 0 \\ c_i + c_i + c_0 & \text{otherwise} \end{cases}$
- Eventually output $c_{\lambda_0}$.

This implementation requires at most $2\lambda_0$ homomorphic additions and exhibits a latency of $\lambda_0$. When querying the evaluation oracle using either the evaluation function $g_t(\mathbf{c})$ or $g'_t(\mathbf{c})$ with $\mathbf{c}$ as a $t$-fold repetition of $c_0$, the resulting ciphertexts are identical, while the amount of computation needed to evaluate $g'_t$ is greatly reduced.

## 3.2 Average-case and empirical noise estimations are not $\mathsf{KR}^\mathsf{D}$ secure

In the threat model, the adversary may select messages and operations for evaluation. If this did not match with the computation or message distribution used to heuristically determine the error bound, then the security guarantee wrt. the Li-Micciancio attack can be destroyed.

The error bound should carefully record the real worst-case bound that can be achieved in the computation; otherwise, the adversary could fool the user. For instance, the in-trusted user may pick a different message and different evaluation circuit to lead to a much larger real error than the estimated error in the decryption client.

In [13], Costache et al. presented an average-case noise analysis for the CKKS scheme using the central limit theorem (CLT). The authors discussed that their approach is suitable for the noise-flooding countermeasure. For example, they wrote in [13] that

> "Not only does our work provide tighter bounds for $B_{\mathsf{ctxt}}$ (resolving an open problem in [13]), but also our analysis enables us to directly characterise the distribution of e and its variance."

In this subsection, we present a simple key-recovery attack showing that the average-case approach can be problematic when being applied in the mode releasing the decryption results.

**The CLT noise estimation.** In such a scenario, the precision tracker needs to make some heuristic assumptions such that the added or multiplied random variables are independent. Then, the tracker can get tighter noise bound using the central limit theorem. For instance, the addition of two ciphertexts will lead to a new ciphertext with noise variance the sum of the noise variances of the two input ciphertexts. In the worst-case analysis, we add the standard deviation instead.

The average-case precision tracker can be more accurate in the average case; the adversary, however, may be able to pick inputs to falsify the assumed independence assumption and make the average-case estimation inaccurate.

We consider the circuit of adding $t$ ciphertexts. In the average case, adding $t$ independent ciphertexts with noise $\varepsilon_i \sim \mathcal{N}(0, \sigma I_n)$ will lead to new ciphertexts with noise $\varepsilon \sim \mathcal{N}(0, \sqrt{t}\sigma I_n)$. If the adversary picks $t$ identical ciphertexts, then the noise distribution follows $\varepsilon \sim \mathcal{N}(0, t\sigma I_n)$.

Importantly, the adversary can efficiently execute $g_t(\mathbf{c})$ by employing $g'_t(\mathbf{c})$, where $\mathbf{c}$ represents a $t$-fold repetition of a ciphertext $c_0$. When using a CLT noise tracker that assumes the independence of the two inputs in a homomorphic addition gate, one can still approximate the output ciphertext's estimated noise as $\varepsilon \sim \mathcal{N}(0, \sqrt{t}\sigma I_n)$. The actual noise distribution similarly adheres to $\varepsilon \sim \mathcal{N}(0, t\sigma I_n)$.

**The empirical noise estimation.** Both the worst-case noise estimation and the CLT noise estimation can be designed so that a ciphertext inherently carries a noise bound, which homomorphic evaluations then automatically update. An alternative approach is what we term 'empirical noise estimation', where the user holding the secret key, or other involved parties, conduct preliminary experiments to set a noise bound for decryption with sharing. However, this empirical method generally presupposes that an adversary must perform certain computations, thereby contradicting the IND-CPA$^\mathsf{D}$ adversary model, which allows the choice of the evaluation function. Additionally, the user in possession of the secret key may lack prior knowledge of the function to be evaluated, as could occur in cases involving private circuits. This lack of information renders the assumptions underlying empirical noise estimation invalid.

Estimating empirical noise can compromise noise-flooding protection due to the formal adversary model in IND-CPA$^\mathsf{D}$

[4]Here, we present the case for arbitrary $t$ to provide a more general perspective on the attack. However, it's worth noting that in practical applications, one would likely choose $t$ to be a power of 2. As a result, the function $g'_t$ becomes considerably simpler, as it merely calculates $c_{i+1} \leftarrow c_i + c_i$.

or $\mathsf{KR}^{\mathsf{D}}$ security, which allows the attacker to select valid ciphertexts and perform computations on them. Furthermore, the distribution of ciphertexts and computation gates significantly influences empirical noise estimation. The attacker can inflate the actual noise beyond the estimated noise by choosing identical ciphertexts as inputs to the simple $g_t()$ function and selecting a much larger value of $t$ than what was used in the empirical noise estimation.

To be specific, the noise bound predetermined by the user possessing the secret key for decryption essentially acts as a constant noise level for the adversary who selects the $t$ value in the $g_t()$ function. Utilizing the attack strategy outlined in Section 3.1, a passive adversary may choose a sufficiently large $t$ value, thereby rendering the noise negligible.

### 3.2.1 Experiments on average-case noise estimation

In this section, we present the experimental results on the OpenFHE library, focusing on average-case noise estimation.

**The noise estimation in OpenFHE.** OpenFHE suggests performing the computation in two steps. First, the noise is estimated on a freshly generated key pair with messages chosen from a suitable set of messages. The noise estimation is done by measuring the noise/precision-loss in the imaginary slots of the decrypted plaintext [13]. In this step, the value $\sigma_{\text{est}}$ is computed. During the actual computation, the decrypted message is subjected to noise flooding with discrete Gaussian noise with a standard deviation of $\sigma_2 = \sqrt{12q}2^{\nu/2}\sigma_{\text{est}}$. Here, $q$ represents the number of decryption queries ($q = 1$ in our attack), and $\nu \geq 30$ is the statistical security parameter.

The split of computation in two steps signifies OpenFHE's use of an empirical noise estimation technique. Concurrently, OpenFHE intends to introduce a feature akin to HElib's noise estimator, which associates a noise bound with each ciphertext and adjusts it dynamically during homomorphic operations (refer to [4]). This embedded noise bound in individual ciphertexts serves as an effective mechanism to reconcile disparities between predicted and actual noise levels that could arise due to a different adversary-chosen evaluation function.

OpenFHE recommends for the use of its own empirical noise evaluation strategy, initially suggesting a method akin to average-case noise estimation. The results from this method are, in specific cases under scrutiny, the same as those acquired through the CLT noise estimation method. Specifically, this involves gauging the noise or precision-loss in the imaginary slots of the decrypted plaintext [13]. In the discussion that follows, we deliberately apply identical computations in both phases, to negate the impact of function variations chosen by an adversary. Our attention is specifically geared towards OpenFHE's average-case noise estimation, highlighting the disparities arising from the input message distributions for simple gates that an adversary could select.

| | $n$ | $\log q$ | $\nu$ | precision | plaintext slots |
|---|---|---|---|---|---|
| OpenFHE | $2^{14}$ | $155-181$ | 30 | 25 | 16 |

Table 1: Parameters used in the attack on OpenFHE. The modulus $q$ is chosen by OpenFHE dynamically, it increases for higher values of $t$; range for $q$ is presented for the values $46 \leq \log t \leq 57$ from the Fig. 3. The value $q$ is allowed to grow up to around $2^{410}$ for the scheme to provide 128 bits of (quantum) security [3].

In Section 3.2.2, we demonstrate that OpenFHE's primary recommendation of empirical noise estimation is more vulnerable to attacks. Specifically, this vulnerability arises because the adversary has the freedom to select a different evaluation function.

**Key recovery attack on OpenFHE.** The attack scenario follows Section 3.1. The user sets up the parameters for the scheme that are provided in Table 1 and generates public and private keys. Then he estimates the noise of $g_t$ for another freshly generated key pair for the adversary's choice of $t$ and initializes the oracles. The adversary then uses encryption and evaluation oracles to compute a ciphertext from $g_t(\mathbf{c})$ and calls decryption oracle on it. The function $g_t(\cdot)$ is implemented as $g'_t(\cdot)$ by both the user and the adversary.

For the $g_t()$ function, the estimated noise $\sigma_{\text{est}}$ is very close to $\sqrt{t}\sigma_{1,\text{est}}$, where $\sigma_{1,\text{est}}$ is the noise estimation for a freshly generated ciphertext. We notice a possible bug in OpenFHE, $\sigma_{1,\text{est}}$ is close to $3.19\sqrt{2/3 \cdot n}$ for the secret with uniform ternary coefficients, whereas the actual noise $\sigma_1$ of a fresh ciphertext is close to $3.19\sqrt{4/3 \cdot n}$. This implies that the injected noise is somewhat less than expected. Though this has a minor impact on the scheme's security, making the attack slightly more effective, the attack can still proceed without this difference.

Treating $\nu$ as a constant, from the formula for the standard deviation of the noise flooding $\sigma_2 = \sqrt{12q}2^{\nu/2}\sigma_{\text{est}}$, we obtain $\sigma_2 = \Theta(\sqrt{t}\sigma_1)$. As mentioned earlier, by choosing a sufficiently large $t$, we can make $\sigma_{\text{attack}}$ negligible. In order to obtain the same value of $\sigma_{\text{attack}}$ for scenarios when $\nu = 0$ and $\nu > 0$, one need to increase $\log t$ by $\nu$. To evaluate $g'_t(\cdot)$, one needs no more than $2\lceil \log t \rceil$ ciphertext additions which translates to $2\lceil \log t \rceil n$ additions over modulo $q$. In other words, the attack complexity linearly depends on $\nu$, which means that by simply increasing $\nu$ the attack could not be prevented.

Let us emphasize once again that the attack relies on a single decryption call. The adversary decrypts $t \cdot c_0$, obtains $e_{\text{total}}$, and computes $b'$. In the real attack scenario, the adversary can utilize $b'$, $a$, and an estimate of $e'$ to extract information about $s$. However, in our experiments, we employ $s$ to derive $e'$ from $b'$ and validate the theoretical estimates.

Figures 3 and 4 demonstrate $\sigma_{\text{attack}}$ and the weight of $e'$, respectively. These figures clearly illustrate that the parameter $\nu$ delays the attack, while the value of $\sigma_{\text{attack}}$ remains the same. The parameters for $\nu = 0$ are the same as in Table 1 except that $\log q$ is lower. When $\log t - \nu \geq 27$, the trivial attack of computing $s = a^{-1}b'$ becomes possible, as $e'$ becomes 0 with a high probability. However, the retrieval of $s$ is not limited to this trivial attack. Even with smaller values of $t$, we can still obtain an RLWE sample with noise smaller than that used in the public key and retrieve the secret key with additional post-processing.

Note that since $e'$ is sampled from a discrete Gaussian distribution, it is natural for the empirical standard deviation to be smaller than the theoretical one for small $\sigma_{\text{attack}}$. This effect can be observed in Fig. 3 when $\log t - \nu \geq 24$. A slight deviation from the theoretical prediction in the preceding part of the graph could be attributed to the fact that Lemma 2 was originally formulated for the continuous Gaussian distribution, whereas we employ the discrete Gaussian distribution, which is approximated by the continuous one.

OpenFHE supports both 64-bit and 128-bit precision computations and uses 64-bit computation by default. Since for the attack to work we need, say, $t = 2^{57}$, the noise grows significantly and to support that we need to compile the library with NATIVE_SIZE $= 128$ option to support 128-bit precision computations.

Experiments were run on a computer with an Intel i7-11700K processor running at 3.60GHz using 64 GB of RAM. For the trivial attack with $\log t = 57$ the running time of all oracles called by the adversary is 2.7 seconds.

After oracle calls the adversary need to compute $b'$ as in Eq. (2), then he can obtain $s = a^{-1}b'$. For simplicity this step was implemented in SageMath, but the computation is not optimized. The total time is over one minute, where most of the time is spent on computing $a^{-1}$.

With values of $\sigma_{\text{attack}}$ approaching 0, the coefficients of $e'$ are usually zeroes with a few coordinates being $-1$ or 1. Outside of lattice reduction techniques, there is a simple post-processing idea, when one assumes that $e'$ has $i$ non-zero coefficients and brute-forces all possible $\binom{n}{i}2^i$ errors $e''$ with $i$ non-zero coefficients from $\{-1, 1\}$ for $i = 0, 1, \ldots$ Then the adversary can compute $s' = a^{-1}(b' - e'')$ and check if the result is small (i.e. ternary for OpenFHE) and, if necessary, try to decrypt a ciphertext with a known message to avoid the false-positive result. Fig. 5 demonstrates the success probability of the attack with and without simple post-processing, the data is the same as for Figs. 3 and 4, i.e. we take empirical probability from 100 runs. The number of guesses for $e''$ is $\sum_{i=0}^{2} \binom{n}{i} 2^i \approx 2^{29}$, making the computation time reasonable. Guessing more positions can significantly increase the post-processing time.

**Summary.** When we focus on the parameter set outlined in Table 1 and choose $t = 2^{57}$, we observe an empirical success rate of 100% (100 successes in 100 trials) without the need for any post-processing. On our desktop system, the average time needed to produce the desired ciphertext to query to the decryption oracle is approximately 2.7 seconds, while the key recovery process takes up to one minute after receiving the decryption results. While it's possible to lower $t$ to $2^{56.5}$ if reasonable post-processing is employed, this modification has a minimal impact on the complexity and latency involved in creating the designed ciphertext. However, it considerably increases the complexity of key recovery. As a result, we recommend adhering to the straightforward version of the attack that excludes post-processing.

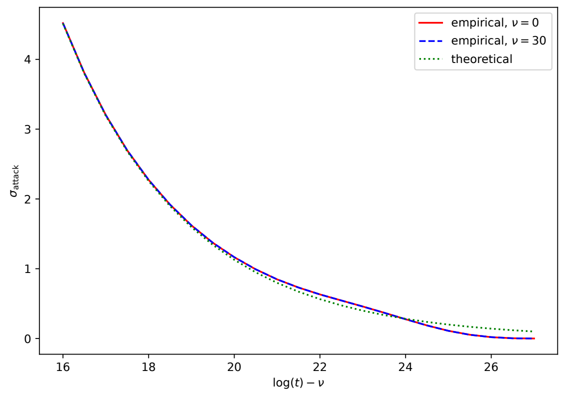


Figure 3: Standard deviation of $e'$. For empirical measurements we take average among 100 runs. The theoretical value is determined from Lemma 2 with continuous Gaussian distribution.

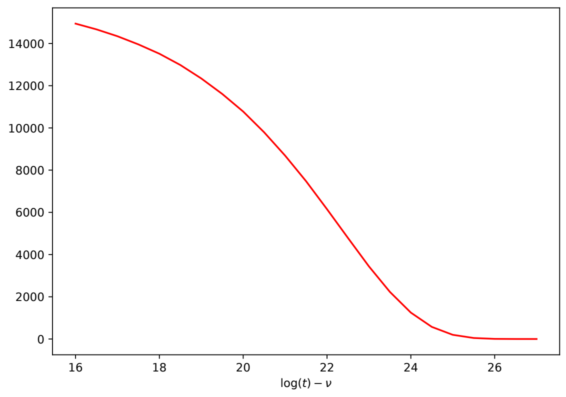


Figure 4: Average weight of $e'$ among 100 runs.

### 3.2.2 Experiments on the empirical noise estimation

We now shift our attention to empirical noise estimation as recommended by OpenFHE. The attack scenario is the same as in Section 3.2.1, but now we assume that the user with secret key estimates the noise for $g_t(\cdot)$ for some small $t$, say $t = 15$, expecting the server to do a few additions. The ad-

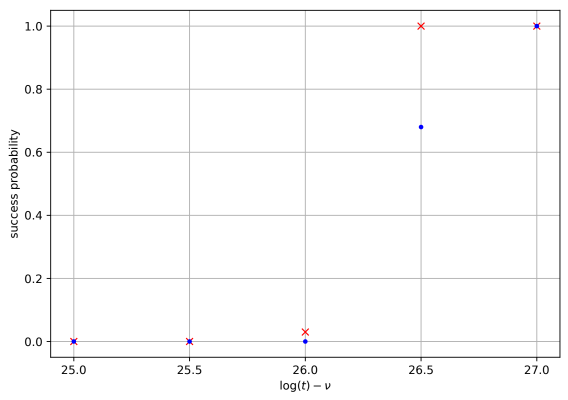


Figure 5: Empirical probability of successful key recovery attack: each data point based on 100 runs. Blue dots correspond to the case without post-processing, i.e. when the noise $e'$ is zero, while red crosses show the probability of obtaining $e'$ with weight less or equal 2, which allows for simple post-processing.

versary, however, computes $g_t(\cdot)$ for $t = 2^{45}$ implementing $g_t$ via $g'_t$. Upon receiving the decryption result, the adversary can effortlessly recover the secret $s$ by computing $s = a^{-1}b'$. Note that the switch from average-case to empirical noise estimation in the described scenarios allows the adversary to reduce $t$ from $2^{57}$ to $2^{45}$ for the straightforward attack without post-processing when targeting parameters from Table 1.

The required value of $\log t$ increases with the value of the statistical parameter $\nu$. The scheme is using $\nu = 30$, therefore, even if we assume that the estimated noise is low, the starting value for $t$ remains relatively high. Despite this, the attack can be more efficient than those discussed in Section 3.2.1. With the same computer platform, the adversary still requires less than three seconds to prepare the ciphertext for decryption and about one minute for key recovery.

If we consider HElib, the attack described in Section 3.2.1 is not successful. That is because HElib keeps track of the error bound for the ciphertext. Most importantly, the error bound estimation employs worst-case analysis. This means that the noise after $g_t$ with the same ciphertexts and the added noise during decryption have the similar standard deviation. However, the documentation of HElib suggests the usage of the function ctxt.bumpNoiseBound in case when the estimated noise is much larger than the real one, determined during the experiments. In other words, the usage of this function can transform the worst-case noise estimation into empirical noise estimation, thus potentially making the mentioned attack possible. We warn about the dangers of such function and relying on the average-case estimations since an attacker may not necessarily adhere to the underlying assumptions of the estimation.

## 4 On deterministic Gaussian noise for fresh ciphertext

In the conclusion section of [23], it was suggested to limit the number of decryption calls to 1 per ciphertext by simply replacing the seed of the pseudorandom function for the additional noise with the ciphertext and key derived from the decryption key. They claim that this proposed countermeasure can prevent an attack where the adversary makes the noise arbitrarily small by using a sufficiently large (still polynomial) number of calls for decrypting the same ciphertext. This countermeasure has been implemented in the HElib library. The rounding protection employed in Lattigo can be regarded as a variant of adding deterministic noise. In [9], Cheon, Hong, and Kim made similar claims, stating that "..an adversary can query the decryption only for fresh ciphertexts, one may choose much smaller additional noise..". However, we will now demonstrate that the countermeasure involving deterministic noise generation does not provide additional security.

**The new idea.** Our novel observation reveals that the secret noise $e$ of a ciphertext $c$ may be exposed through querying decryption outputs of ciphertexts as a function incorporating $c$, rather than solely through the decryption outputs of the exact ciphertext $c$. Consequently, we generate multiple ciphertexts derived from $c$ and obtain corresponding decryption outputs to gather information about the secret noise $e$. For example, for a ciphertext $c$ which is an encryption of all zeros, we submit ciphertexts of $t \cdot c$ for decryption, with varying values of $t$. This approach yields the noise $t \cdot e + \varepsilon(t \cdot c, s)$ for a specific integer $t$. As $t$ changes, the pseudorandom generator's seed alters. The error $\frac{\varepsilon(t \cdot c, s)}{t}$ remains random, ultimately allowing for the acquisition of multiple approximate samples of $e$ to minimize the additional noise to an arbitrarily small amount.

The ciphertexts $t \cdot c$ can be derived by homomorphically evaluating the scalar multiplication of $c$ by $t$ or the addition gate of $g_t(c, \dots, c)$. As detailed in Section 3, estimating the noise size of $g_t(\cdot)$ using an average-case approach enables a more efficient attack on the scheme. We now illustrate that even with worst-case noise size estimation, incorporating deterministic noise for fresh ciphertexts fails to provide extra protection. A prime example is the current HElib implementation; when worst-case noise estimation with no empirical noise setting is employed, the standard deviation of $\varepsilon(t \cdot c, s)$ is equal to the one of $t \cdot e$, and, subsequently, the error $\frac{\varepsilon(t \cdot c, s)}{t}$ exhibits the same standard deviation as $e$. From this point forward, we assume that noise is estimated in the worst-case manner and modify our notation accordingly.

As described in Algorithm 1, we denote $c_0 := (a, -as + e)$ a ciphertext encrypting zero, and for $i > i_0$, $c_i = (i \cdot a, -i \cdot as + i \cdot e)$ a fresh ciphertext of zero, which is the homomorphical multiplication of $c_0$ by $i$. With a raw decryption query of $c_i$,

**Algorithm 1** Key recovery attack on noise-flooding countermeasure with deterministic noise generation.

```
1: Input: Lattice parameters $(n, \log q)$, initial scaling factor $\Delta_0$
2: Sample $(\mathsf{sk}, \mathsf{pk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}(n, \log q, \Delta_0)$
3: Encrypt $c_0 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mathsf{Encode}(\mathbf{0}; \Delta_0))$
4: $\hat{z} \leftarrow \mathbf{0}$
5: for $i_0 \leq i < \gamma + i_0$ do
6:     Homomorphically evaluate $c_i \leftarrow g_i(c_0, c_0, \ldots c_0)$
7:     Decrypt $z_i \leftarrow \mathsf{DecForShare}_{\mathsf{sk}}(c_i)$
8:     $z_i \leftarrow \frac{1}{i} z_i$
9:     $\hat{z} \leftarrow \hat{z} + \frac{1}{\gamma} z_i$
10: end for
11: $m' \leftarrow \mathsf{Encode}(\hat{z}; \Delta_0)$
12: Compute $b' \leftarrow m' - b \in R_q$, where $(a, b) = c_0$    ▷ $b' \simeq a \cdot s + X_\gamma$
13: if POST-PROCESSING $= 1$ then
14:     Return $s' := \mathsf{PostPro}(a, b')$, where $\mathsf{PostPro}(\cdot)$ represents a post-processing procedure, which could be enumeration or lattice reduction methods
15: else
16:     Return $s' := a^{-1} \cdot b'$ for the trivial attack when $X_\gamma = 0$ with high probability
17: end if
```

an adversary can passively get $i \cdot e$. If we add a noise $\varepsilon(c_i, s)$ issued from a pseudorandom generator applied to $(c_i, s)$ to hide this error, the attacker will get an approximation of $i \cdot e + \varepsilon(c_i, s)$.

We denote $z_i \leftarrow \mathsf{DecForShare}_{\mathsf{sk}}(c_i)$ the result from the decryption query followed by noise flooding and decoding. From these observations, an attacker can compute $z'_i = \frac{1}{i} z_i$, compute the empirical mean by $\hat{z} = \sum_{i=1}^{\gamma} \frac{1}{\gamma} z'_i$ and encode it to have

$$\mathsf{Encode}(\hat{z}) \simeq \Delta \varphi^{-1} \left( \frac{1}{\gamma} \sum_{i=1}^{\gamma} z'_i \right) \simeq \frac{\Delta}{\gamma} \sum_{i=1}^{\gamma} \varphi^{-1}(z_i) = e + X_\gamma$$

with

$$X_\gamma \simeq \frac{1}{\gamma} \left( \sum_{i=1}^{\gamma} \frac{1}{i} \varepsilon(c_i, s) \right)$$

The value $X_\gamma$ is the sum of independent centered Gaussian noises of the same standard deviation so it behaves as a centered Gaussian noise with a smaller variance. Doubling $\gamma$ divides the variance by two. To have access to $\eta$ bits of $e$ with overwhelming probability, the attacker will only need to have $\gamma = \Theta(\eta^2)$ decrypted results.

Contrastingly, without applying the countermeasure of deterministic noise generation for fresh ciphertexts, each query introduces a fresh noise with a standard deviation equal to that of $e$, equivalent to adding noise of $\frac{\varepsilon(i \cdot c, s)}{i}$. To recover $\eta$ bits of the secret $e$, we still need $\gamma = \Theta(\eta^2)$ decrypted outputs.

In summary, the new method illustrates that employing deterministic noise for fresh ciphertexts offers no additional protection.

|  | $n$ | $\log q$ | precision | plaintext slots |
|---|---|---|---|---|
| HElib | $2^{13}$ | 108 | 20 | $2^{12}$ |

Table 2: The targeted parameters used in the experiments on HElib.

### 4.1 Application to HElib

We implemented the attack described in Algorithm 1 for HElib. Secret key coefficients in this library are sampled from the ternary set, where the probability to sample 0 is $1/2$, and the rest probability mass is equally split between 1 and $-1$. Consequently, the expected noise of a fresh ciphertext is given by $\sigma_1 = 3.2\sqrt{n+1}$. Since HElib uses worst-case estimation for the noise and due to canonical embedding we have $\sigma_{1,\mathsf{est}} \simeq 3.2(\sqrt{2n \ln n} + 1)$. At the end of the algorithm the standard deviation of $X_\gamma$ is approximately

$$\sigma_{\mathsf{attack}} = \frac{\sigma_{1,\mathsf{est}}}{\sqrt{\gamma}}.$$

For our experiments, we selected parameters according to Table 2 to ensure a desired level of security, specifically, the overall security level is more than 128 bits, as outlined in [3]. We compared the theoretical value of $\sigma_{\mathsf{attack}}$ with the actual values obtained during our experiments. Fig. 6 presents a graphical representation of these results. It is evident from the figure that our experimental data aligns closely with the theoretical predictions, validating the correctness of the results in Section 4.

We provide an overview of the implications of our attack on the HElib library. Given that HElib has not implemented any updates incorporating the countermeasure through differential privacy [24], it lacks IND-CPA$^{\mathsf{D}}$ secure. On the other hand, attack strategy detailed in this section requires millions of decryption queries to successfully retrieve the secret key when the worst-case noise bound for each ciphertext is dynamically updated. Consequently, when noise estimation is executed in a worst-case manner, the encryption scheme maintains its KR$^{\mathsf{D}}$ security. Last, the attack discussed in Section 3.2.2 could work when targeting implementations with the empirical noise estimation method of invoking the function ctxt.bumpNoiseBound to manually adjust the noise level utilized in the decryption function for sharing.

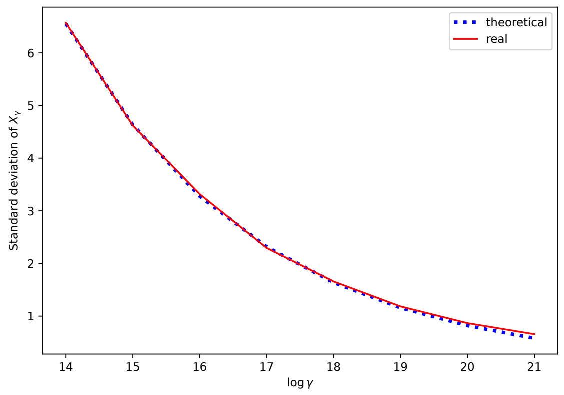


Figure 6: Standard deviation of $X_\gamma$ according to the prediction and determined experimentally.

## 5 Concluding remarks

In this paper, we have proposed new key-recovery attacks on Approximate Homomorphic Encryption schemes that use noise-flooding countermeasures based on non-worst-case noise estimation, which are implemented to attain IND-CPA$^{\mathrm{D}}$ security. We illustrated the attack process using the recent implementation of the OpenFHE library and showed how to recover the secret key with a single decryption output. Furthermore, we discussed the security implications of our attack on the HElib implementation, which allows the use of empirical noise estimation. As an additional finding, we revealed that using a pseudorandom generator to create deterministic noise in noise-flooding countermeasures does not enhance security.

**Mitigation and future works.** Our research underscores the importance of employing worst-case noise estimation in noise-flooding countermeasures. By adopting the security bounds proposed in [24], which stem from differential privacy, and implementing a worst-case noise estimation strategy, our attack can be thwarted. Nevertheless, the application of such countermeasures can lead to a considerable reduction in performance, emphasizing the need to investigate more efficient countermeasures in future work.

Although one could formulate defenses against our attacks that employ the function $g_t(\cdot)$ – for instance, by limiting the acceptable average-case noise bound for ciphertexts, thus disallowing large $t$ values conducive to key recovery – such countermeasures often incur significant performance costs. Moreover, they may prove ineffective against adversaries who craft new evaluation functions for emerging passive attacks. It remains an intriguing question as to whether we can develop a more efficient implementation using average-case noise estimation, while still preserving the scheme's IND-CPA$^{\mathrm{D}}$ security (or perhaps only the KR$^{\mathrm{D}}$ security).

## Acknowledgement

This work was supported by the Swedish Research Council (grant numbers 2019-04166 and 2021-04602); the Swedish Civil Contingencies Agency (grant number 2020-11632); the Swedish Foundation for Strategic Research (Grants No. RIT17-0005 and RIT17-0032); and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The computations and simulations were partly enabled by resources provided by LUNARC.

## References

[1] Heaan. https://github.com/snucrypto/HEAAN.

[2] PALISADE Lattice Cryptography Library (release 1.11.5). https://palisade-crypto.org/, September 2021.

[3] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018.

[4] Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Ian Quah, Yuriy Polyakov, Saraswathy R.V., Kurt Rohloff, Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. Openfhe: Open-source fully homomorphic encryption library. Cryptology ePrint Archive, Paper 2022/915, 2022. https://eprint.iacr.org/2022/915.

[5] Fabian Boemer, Yixing Lao, Rosario Cammarota, and Casimir Wierzynski. ngraph-he: a graph compiler for deep learning on homomorphically encrypted data. In *Proceedings of the 16th ACM International Conference on Computing Frontiers*, pages 3–13, 2019.

[6] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press.

[7] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In Jesper Buus Nielsen

and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 360–384, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[8] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full RNS variant of approximate homomorphic encryption. In Carlos Cid and Michael J. Jacobson Jr:, editors, *SAC 2018: 25th Annual International Workshop on Selected Areas in Cryptography*, volume 11349 of *Lecture Notes in Computer Science*, pages 347–368, Calgary, AB, Canada, August 15–17, 2019. Springer, Heidelberg, Germany.

[9] Jung Hee Cheon, Seungwan Hong, and Duhyeong Kim. Remark on the security of CKKS scheme in practice. Cryptology ePrint Archive, Report 2020/1581, 2020. https://eprint.iacr.org/2020/1581.

[10] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany.

[11] Jung Hee Cheon, Duhyeong Kim, and Jai Hyun Park. Towards a practical cluster analysis over encrypted data. In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019: 26th Annual International Workshop on Selected Areas in Cryptography*, volume 11959 of *Lecture Notes in Computer Science*, pages 227–249, Waterloo, ON, Canada, August 12–16, 2019. Springer, Heidelberg, Germany.

[12] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 3–33, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

[13] Anamaria Costache, Benjamin R. Curtis, Erin Hales, Sean Murphy, Tabitha Ogilvie, and Rachel Player. On the precision loss in approximate homomorphic encryption. Cryptology ePrint Archive, Report 2022/162, 2022. https://eprint.iacr.org/2022/162.

[14] Roshan Dathathri, Olli Saarikivi, Hao Chen, Kim Laine, Kristin Lauter, Saeed Maleki, Madanlal Musuvathi, and Todd Mytkowicz. Chet: an optimizing compiler for fully-homomorphic neural-network inferencing. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 142–156, 2019.

[15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 617–640, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[16] EPFL-LDS. Lattigo 2.2.0. https://github.com/tuneinsight/lattigo.

[17] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. https://eprint.iacr.org/2012/144.

[18] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.

[19] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[20] Kyoohyung Han, Seungwan Hong, Jung Hee Cheon, and Daejun Park. Logistic regression on homomorphic encrypted data at scale. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 9466–9471, 2019.

[21] IBM. Helib. https://github.com/homenc/HElib.

[22] Joohee Lee, Dongwoo Kim, Duhyeong Kim, Yongsoo Song, Junbum Shin, and Jung Hee Cheon. Instant privacy-preserving biometric authentication for hamming distance. Cryptology ePrint Archive, Report 2018/1214, 2018. https://eprint.iacr.org/2018/1214.

[23] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 648–677, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany.

[24] Baiyu Li, Daniele Micciancio, Mark Schultz, and Jessica Sorrell. Securing approximate homomorphic encryption using differential privacy. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 560–589, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany.

[25] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

[26] WA Microsoft Research, Redmond. Microsoft seal. https://github.com/microsoft/SEAL.

[27] Saerom Park, Jaewook Lee, Jung Hee Cheon, Juhee Lee, Jaeyun Kim, and Junyoung Byun. Security-preserving support vector machine with fully homomorphic encryption. *SafeAI@ AAAI*, 2301, 2019.

[28] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[29] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.