Attacks Against the IND-CPA^D Security of Exact FHE Schemes

Jung Hee Cheon
Seoul National University & CryptoLab Inc.
Seoul, Republic of Korea
jhcheon@snu.ac.kr

Hyeongmin Choe
Seoul National University
Seoul, Republic of Korea
sixtail528@snu.ac.kr

Alain Passelègue
CryptoLab Inc.
Lyon, France
alain.passelegue@cryptolab.co.kr

Damien Stehlé
CryptoLab Inc.
Lyon, France
damien.stehle@cryptolab.co.kr

Elias Suvanto
CryptoLab Inc. & University of Luxembourg
Lyon, France & Luxembourg
elias.suvanto@cryptolab.co.kr

Abstract

A recent security model for fully homomorphic encryption (FHE), called IND-CPA^D security and introduced by Li and Micciancio [Eurocrypt'21], strengthens IND-CPA security by giving the attacker access to a decryption oracle for ciphertexts for which it should know the underlying plaintexts. This includes ciphertexts that it (honestly) encrypted and those obtained from the latter by evaluating circuits that it chose. Li and Micciancio singled out the CKKS FHE scheme for approximate data [Asiacrypt'17] by giving an IND-CPA^D attack on it and claiming that IND-CPA security and IND-CPA^D security coincide for exact FHE schemes.

We correct the widespread belief according to which IND-CPA^D attacks are specific to approximate homomorphic computations. Indeed, the equivalency formally proved by Li and Micciancio assumes that the schemes have a negligible probability of incorrect decryption. However, almost all competitive implementations of exact FHE schemes give away strong correctness by analyzing correctness heuristically and allowing noticeable probabilities of incorrect decryption.

We exploit this imperfect correctness to mount efficient non-adaptive indistinguishability and key-recovery attacks against all major exact FHE schemes. We illustrate their strength by implementing them for BFV using OpenFHE and simulating an attack for the default parameter set of the CGGI implementation of TFHE-rs (the attack experiment is too expensive to be run on commodity desktops, because of the cost of CGGI bootstrapping). Our attacks extend to CKKS for discrete data, and threshold versions of the exact FHE schemes, when the correctness is similarly loose.

Keywords: Fully homomorphic encryption; IND-CPA^D security

1 Introduction

In a fully homomorphic encryption (FHE) scheme, arbitrary circuits can be publicly applied to encrypted data. The most direct application is to outsource computations confidentially: a client encrypts data and sends the ciphertexts and a circuit to the server; given the ciphertexts, the server homomorphically evaluates the circuit and produces a new ciphertext that it sends back to the client; the client then decrypts to obtain the result of the computation. If the FHE scheme is indistinguishable under chosen plaintext attacks (IND-CPA), then the server cannot learn anything about the client's data. Application scenarios of FHE and its extensions abound, and additional security properties may be required [40].

IND-CPA security suffices for applications of FHE in which the data that shall remain confidential is accessible only to the entity that encrypts it and to the owner of the decryption key. In this work, we are interested in a different type of applications of FHE where the decrypted data is being shared publicly. Let us consider three parties: the encryptor, the evaluator, and the decryptor. A typical scenario is the case in which an entity (the decryptor) is willing to allow for some computation (e.g., statistics) over its data to a client (the encryptor) via a public server performing the computation (the evaluator). FHE allows to do so without requiring the entity to reveal the raw data to the server nor the client. The adversary is the encryptor. It is honest but curious, in the sense that it encrypts by following the specifications of the encryption procedure. It can then request the evaluator to homomorphically evaluate circuits of its choice, and can ask the decryptor to decrypt a ciphertext that is produced honestly via encryption and/or evaluation. To address this scenario, Li and Micciancio [37] extended IND-CPA security to the notion of indistinguishability under chosen plaintext attacks with a decryption oracle (IND-CPA^D). In the security game, the adversary is given access to the above-mentioned oracles, and should not be able to learn information about the decryptor's data, up to what is revealed by the result of the computation. It is formalized by indistinguishability of chosen (challenge) plaintexts with the restriction that the

decryption oracle can be called only on ciphertexts whose underlying plaintexts are independent of the challenge bit.

Note that, while IND-CPA$^{\mathsf{D}}$ security might not be needed for many applications involving a single decryptor, a very similar security notion is necessary in the context of threshold FHE [6, 10], which has numerous applications (e.g., it is used in smart contracts [23]). In the threshold setting, the secret key is shared between distinct parties (decryptors) and decryption is done by involving multiple decryptors: each of them partially decrypts a ciphertext using its share of the key, and the plaintext is obtained by recombining all partial decryptions. A decryptor revealing its partial decryption is analogous to making a decryption query in the single decryptor setting, thus IND-CPA$^{\mathsf{D}}$ security can be seen as the simplest form of threshold security (i.e., when a single decryptor owns the actual secret key). In the rest of the paper, we focus on IND-CPA$^{\mathsf{D}}$ security and conclude by extending our contributions to the threshold setting.

At first sight, one may think that the public sharing of the decryption should not help the adversary in any way, as decryption should provide information that it already has. The definition from Li and Micciancio is motivated by the CKKS FHE scheme [18], which performs arithmetic computations on approximations to real/complex plaintexts. In this scheme, the errors due to inaccuracies of the approximations are entangled with the errors introduced for security, due to the use of the Learning With Errors problem (LWE) [43]. Decryption then gives information on the latter type of errors. In fact, simply encrypting a message and requesting its decryption already gives a linear equation with the secret key vector. By repeating this, the adversary obtains an invertible system of linear equations and can recover the secret key. This gives a key-recovery with a decryption oracle (KR$^{\mathsf{D}}$) attack. KR$^{\mathsf{D}}$ security, later defined in [38], is an adaptation of IND-CPA$^{\mathsf{D}}$ security that asks the attacker to recover the secret key. It is a weaker notion of security than IND-CPA$^{\mathsf{D}}$ security, as a KR$^{\mathsf{D}}$ attack gives an IND-CPA$^{\mathsf{D}}$ attack. Li and Micciancio showed that "exact" FHE schemes are immune to IND-CPA$^{\mathsf{D}}$ attacks, which leads them to conclude that the IND-CPA$^{\mathsf{D}}$ security shows a weakness that would be specific to approximate FHE schemes. This leads to the current state of affairs: it is widely believed that only CKKS is susceptible to IND-CPA$^{\mathsf{D}}$ attacks, while other ("exact") schemes are immunized.

This common belief is due to misleading terminology in [37]. The word "exact" is used to oppose older, "standard", FHE schemes such as BGV [15], BFV [14, 27], DM [26], and CGGI [19] that operate on exact data, to the "approximate" CKKS FHE scheme, which supports real/complex data with approximate homomorphic evaluation. It is assumed in [37] that an exact scheme is correct. The definition of correctness in [37, Section 2] for schemes operating on exact data states that the decryption failure probability should be negligible in the security parameter, i.e., asymptotically smaller than any inverse polynomial. For standard schemes focusing on IND-CPA security (even mere public-key encryption algorithms), this translates in practice to values that are sufficiently small to not alter functionality, such as 2^{-40} or 2^{-60}. However, when the correctness can be adversarially exploited, as in IND-CPA$^{\mathsf{D}}$ security, the decryption error probability should be set exponentially small in the security parameter (e.g., 2^{-128}). As a result, a "standard exact" scheme would be correct for a notion of correctness that is only related to functionality, as per the definition in [37, Section 2], but not adversarially correct, as per what would be needed to obtain IND-CPA$^{\mathsf{D}}$ security from [37, Lemma 1]. Setting the decryption failure probability to 2^{-40} instead of 2^{-128} helps performance, and hence a larger failure probability is often preferred as it is (incorrectly) believed that this affects functionality but not security, as "exact" schemes would be immunized to IND-CPA$^{\mathsf{D}}$ attacks.

Our contributions We exhibit non-adaptive IND-CPA$^{\mathsf{D}}$ and KR$^{\mathsf{D}}$ attacks for BGV/BFV, DM/CGGI, disproving the common belief that IND-CPA$^{\mathsf{D}}$ insecurity is specific to approximate schemes. For many available implementations of the aforementioned exact schemes, our attacks are very efficient. We report experimental data showing their practical strength, considering the BFV implementation from OpenFHE [7] and the CGGI implementation from TFHE-rs [47]. We stress that these are only examples, our attack being widely applicable. We also propose IND-CPA$^{\mathsf{D}}$ attacks against recent variants of CKKS operating over discrete data [3, 20, 25]. Our attacks extend to Threshold-FHE, and we discuss the impact of our attack on the Noah's Ark scheme from [24] as an illustration.

The main take-away of our work is that IND-CPA$^{\mathsf{D}}$ and KR$^{\mathsf{D}}$ attacks are not about exact versus approximate schemes, but about decryption correctness. In particular, our attacks highlight that decryption correctness *is not* only a matter of functionality: it is crucial for security as soon as information about the result of the decryption is made public. In particular, most of our attacks do not require the full knowledge of the decryption result but only on partial information about decryption. Notably, for our attack against DM/CGGI and CKKS, it is sufficient to know a single bit of information per ciphertext and per message slot, respectively: whether the decryption was correct or not. Real-life scenarios in which such information could be available[1] thus require strong correctness guarantees.

What do the attacks exploit? Recall that the proof of [37, Lemma 1], which states that IND-CPA security implies

[1]For example, consider a scenario involving homomorphic detection of a rare disease over encrypted health data: a patient receiving a positive test might alert about potential false positive, which could be the result of a failed decryption. Having this information is sufficient for a non-adaptive attacker to mount our key-recovery attacks against DM/CGGI or CKKS.

IND-CPA^D security for exact schemes, relies on the correctness of the FHE scheme, defined in [37, Section 2]. There are two significant caveats when translating this lemma into practice. First, for all efficient implementations of FHE schemes, correctness is heuristic. Statements on correctness even involve probabilities in contexts where there is no randomness, heuristically modeling some errors occurring in homomorphic circuit evaluation as probabilistic. Second, the decryption failure probability may be sufficiently small for honest users but not for adversaries collecting decryption failures: concrete decryption failure "probabilities" can be of the order of 2^{-40} [47] or even higher (for instance, Concrete-python, a TFHE compiler for TFHE-rs, has decryption failure probability of the order of 2^{-17} [46]). Some of our attacks exploit the heuristic aspects of correctness analyses, others exploit the overly high decryption failure probabilities.

A key-recovery attack on BFV/BGV. The BFV/BGV schemes are based on leveled schemes that support homomorphic evaluation of arithmetic circuits with bounded multiplicative depth, and on bootstrapping procedures that turn them in fully homomorphic schemes. In this work, we consider their leveled variants. As they rely on the RLWE [39, 44] problem, their ciphertexts are associated to error (or noise) components. Since the error increases at each homomorphic operation and may interfere with the plaintext when it becomes sufficiently large, bounding the magnitudes of the errors is important to guarantee correctness. Different noise management strategies are available in the literature to ensure the correctness of these schemes. Absolute bounds based on the triangle inequality provide rigorous correctness guarantees but lead to large parameters. To avoid this performance penalty, most current implementations rely on heuristic noise estimates. Some assume that manipulated ciphertexts have noise terms that are Gaussian and independent. These heuristic noise estimates are much closer to what happens in typical executions, but correctness is only heuristic. A recent work [28] showed that those heuristic noise estimates can be exploited to obtain a KR^D attack against CKKS, by using correlated input ciphertexts. We show that such noise bounds also make BFV/BGV insecure by adapting the KR^D attack to the CKKS context. The attack adds an encryption of 0 sufficiently many times with itself so that the noise moves to the plaintext position and can be read by requesting a decryption. We successfully mounted the attack against a BFV implementation based on such error analysis, using OpenFHE. The attack actually breaks the default BFV implementation, but for another reason (see related works, below).

A key-recovery attack on DM/CGGI. In the DM/CGGI FHE schemes, evaluating a circuit gate is performed jointly with a bootstrapping operation. We describe our attack for CGGI, which relies on binary secret key vectors, but note that it can be adapted to DM. Bootstrapping is a procedure that reduces the noise of a given ciphertext, allowing for further homomorphic manipulations. These schemes make use of LWE [43] formats for regular ciphertexts, and GLWE [15, 35] formats inside the so-called BlindRotate procedure. In the BlindRotate procedure, GLWE ciphertexts encrypt powers of X whose exponents correspond to the pre-BlindRotate ciphertext. As a result, the pre-BlindRotate ciphertexts are defined modulo $2N$, where N is the GLWE ring degree. For efficiency purposes N is taken as small as possible, implying the need to switch the modulus q of regular ciphertexts to $2N$ before BlindRotate. This step is called ModSwitch. Our attack focuses on this step, because it introduces significant noise and hence significantly contributes to the bootstrapping error probabilities, and because the rounding noise $\tilde{\mathbf{e}}$ is publicly known. When a decryption failure occurs, we obtain that $\langle \tilde{\mathbf{e}}, \mathbf{s} \rangle + e \geq t$, where $\mathbf{s}$ is the secret key, the term e can be modeled as an independent integer Gaussian sample, and t is a correctness threshold. As we have LWE samples for $\mathbf{s}$, this looks like an instance of LWE with hints, which has been the focus of recent works on lattice-cryptography cryptanalysis [21, 22]. Running the security estimate script corresponding to [22] already gives interesting attack costs, but we take an even more direct approach. We observe that conditioned on a decryption failure, the distribution of $\tilde{e}_i$ is (heuristically) uniform in $[-1/2, 1/2]$ when $s_i = 0$, and very different when $s_i = 1$. We then amplify the distributional discrepancy by considering many decryption failures, allowing us to recover $\mathbf{s}$.

To assess the strength of our attack, we considered the TFHE-rs library [47]. For its default parameters, the bootstrapping failure probability is of the order of 2^{-40}. We then expect the adversary to observe a bootstrapping failure in large enough homomorphic computations such as those processing more than 128 GB $\simeq 2^{40}$ bits of underlying plaintexts. Due to the limited performance of TFHE-rs, computing 2^{40} bootstraps takes around 300 years on a single-thread CPU. To mount our efficient key recovery algorithm, we provide two experiments producing hints that circumvent the cost of bootstrapping: In the first one, we modify the TFHE-rs parameters so that it still has 128 bits of IND-CPA security but with a bootstrapping failure probability of around 1%. In this case, we are able to compute sufficiently many bootstraps to mount our attack in practice using only the TFHE-rs API. In the second experiment, we keep the default TFHE-rs parameters, but simulate bootstrapping errors rather than running bootstrapping itself. In this experiment, we recover all coefficients of the secret key $\mathbf{s}$ with the observation of only 256 bootstrapping failures.

A key-recovery attack on CKKS over discrete data. Recent works introduce techniques for discrete computations using CKKS instead of BFV/BGV or DM/CGGI by discretizing the CKKS message space of CKKS [3, 20, 25]. Another recent

work [31] uses the CKKS bootstrapping algorithm as a subroutine to accelerate the expensive BGV/BFV bootstrapping [31]. However, with some small probability, CKKS bootstrapping fails, in that the plaintexts below the input and output ciphertexts are very far away. As a results, the exact schemes mentioned just above may have significant failure probability due to the CKKS bootstrapping. In the literature, the failure probability of CKKS bootstrapping can vary from 2^{-16} [11] to 2^{-138} [2, 12], but a failure probability of the order of 2^{-35} is often preferred to make bootstrappings more efficient. This is also the case in the discrete computations using CKKS, and thus our KR^{D} attack is applicable. We note that CKKS bootstrapping failures also occur in the CKKS approximate FHE scheme even with noise flooding countermeasures [38], if the bootstrapping failure probability is not sufficiently small. This is notably the case of [7], which claims $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ security, but to which our attack applies.

In CKKS, the post-multiplication rescaling consumes ciphertext modulus and, once it gets too small, no more homomorphic computations are available. The CKKS bootstrapping replenishes the ciphertext modulus and enables further computations. The scheme relies on RLWE formats for ciphertexts, like BGV/BFV: a ciphertext is a pair of elements in $\mathcal{R} := \mathbb{Z}[X]/(X^N + 1)$ modulo a ciphertext modulus, for a power-of-two integer N. When enlarging the modulus from a small q to a larger Q, a ciphertext $(\mathbf{a}, \mathbf{b} = \mathbf{as} + \mathbf{m} + \mathbf{e}) \in \mathcal{R}_q^2$ can be regarded over $\mathcal{R}$: it satisfies $\mathbf{b} = \mathbf{as} + \mathbf{m} + \mathbf{e} + q\mathbf{I}$, for some $\mathbf{I} \in \mathcal{R}$, where $\mathbf{s}$ is a secret key, $\mathbf{e}$ is a small error, and $\mathbf{m}$ is a message. If $Q \gg q$, we can interpret the ciphertext over $\mathcal{R}_Q$ instead of $\mathcal{R}$: it then decrypts to $\mathbf{m} + \mathbf{e} + q\mathbf{I}$. The CKKS bootstrapping applies the modulo q operation homomorphically on the ciphertexts, by evaluating a polynomial approximating the modulo function in the range $[-(K-1)q, (K-1)q]$ for some integer K. Our attack focuses on this approximation range: the polynomial evaluation introduces a large error if the input lies outside the range. When a post-bootstrapping decryption fails, we obtain an inequality $\|\mathbf{b} - \mathbf{as} - \mathbf{e}\|_\infty \geq Kq$. By properly tweaking the attack, we can obtain the coordinate in which the overflow occur, along with its sign. This leads to a secret key recovery attack, similar to the one for DM/CGGI.

We use the Lattigo library [2] to test our attack. For this purpose, we generate custom parameters with 128 bits of IND-CPA security, having high failure probability.

Extension to Threshold-FHE. We complete our work by extending the $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ attacks to threshold FHE schemes. To a threshold-FHE scheme, we associate the (non-threshold) FHE scheme with the same key generation, encryption and evaluation algorithms, and with decryption defined as the composition of the partial decryptions and recombination. If there is an $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ attack on the FHE scheme, it is possible to derive an attack on the threshold-FHE scheme.

Codes of our experiments. The codes are publicly available at the following git repository:

https://github.com/hmchoe0528/INDCPAD_HE_ThresFHE

Vulnerability disclosure. We initiated a discussion involving all libraries we are aware of. Some libraries have updated their guidelines in light of our attacks (see, e.g., recent updates of [7, Section 2.6] and https://github.com/zama-ai/tfhe-rs).

Concurrent works. Recently, Checri et al. [16] proposed an alternative attack against exact FHE schemes. Their attack starts similarly to our KR^{D} attack against BFV (Algorithm 4, itself adapted from [28]). Their adversary queries iterative additions of the ciphertext with itself and its decryption until the obtained ciphertext fails to decrypt. It then begins a dichotomy search to find the smallest integer α such that the ciphertext $\alpha \cdot ct$ (obtained with homomorphic additions) fails to decrypt. It uses the obtained information about the secret LWE error to recover the secret key by Gaussian elimination. We note that this adversary against DM/CGGI can be thwarted by requiring bootstrapping is performed after every addition. In contrast, we present a KR^{D} adversary relying on a weakness of the bootstrapping algorithm. We further note that the strategy from [16] requires adaptive queries while all our attacks are non-adaptive.

Following [16, 28] and a prior version of this work, Alexandry et al. [5] proposed an alternative security notion, termed application-aware $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ security. This notion restricts the capability of the adversary in the $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ game, by requiring it to only encrypt plaintexts and evaluate circuits from restricted sets of inputs and functions corresponding to a target application (we note that such a restriction was already considered in [34, Section 6.2]). The formal definition requires that parameters are set so that decryption correctness holds for all inputs and functions in those sets. Further, practical guidelines to instantiate this requirement are provided. Our BFV attack is claimed to bypass the normal OpenFHE parameter generation mechanism, as it would set scheme parameters for a given application and use the scheme instantiation for another application (see [5, Section 6.2]). This is incorrect: our attack has a single circuit, that we use for both parameter setting and evaluation. However, it should be noted that our attack is analyzed in a setup where the scheme parameters are set assuming that ciphertext noise terms are statistically independent. This aggressive assumption is not made in the OpenFHE parameter generation mechanism. This being said, our attack actually succeeds against OpenFHE even if we follow all guidelines to set parameters. This is not due to a heuristic aspect of the error analysis but to an incorrect worst-case error analysis for additions in OpenFHE. See Section 3.

Paper organization. After reminding preliminary definitions in Section 2, we describe our attacks based on incorrect

noise estimates for BFV/BGV in Section 3. Our attacks exploiting large bootstrapping failure probability (against DM/CGGI and discrete CKKS) are detailed in Section 4. In addition, we describe two additional results in the appendix. First, we provide a generic IND-CPA$^{\mathsf{D}}$ attack against exact schemes with non-zero decryption failure, whose advantage is essentially the decryption error probability multiplied by the number of decryption queries. This attack is explained in Appendix A. Second, we discuss the implications of our attacks to threshold FHE schemes in Appendix B.

2 Preliminaries

Notation. Vectors and polynomials are denoted in bold fonts. The i-th component of a vector or the i-th coefficient of a polynomial is denoted with subscript i. Given a measurable set X, we let $\mathcal{U}(X)$ denote the uniform distribution over X.

For any real $\sigma > 0$ and a vector $\mu \in \mathbb{R}^n$, define the Gaussian function on $\mathbb{R}^n$ centered at μ with standard deviation parameter σ as $\rho_{\mu,\sigma} = \exp(-\|\mathbf{x}-\mu\|^2/2\sigma^2)$ for all $\mathbf{x} \in \mathbb{R}^n$. For the discrete Gaussian distribution over $\mathbb{Z}^n$, with center parameter μ and standard deviation parameter σ, we denote as $D_{\mu,\sigma}$ which is defined by

$$\forall \mathbf{x} \in \mathbb{Z}^n : \frac{\rho_{\mu,\sigma}(\mathbf{x})}{\sum_{\mathbf{x}\in\mathbb{Z}^n} \rho_{\mu,\sigma}(\mathbf{x})} \ .$$

We omit the subscript μ when it is 0.

For the normal distribution over $\mathbb{R}$, centered at μ and with standard deviation σ, we use the notation $\mathcal{N}(\mu,\sigma^2)$. Its probability density is $(1/\sqrt{2\pi}\sigma)\rho_{\mu,\sigma}(x)$ (for $x \in \mathbb{R}$). We let $\mathrm{erfc}(x)$ denote the complementary error function, defined as $\mathrm{erfc}(x) = 1 - (2/\sqrt{\pi}) \cdot \int_0^x \exp(-t^2)\mathrm{d}t$. It then holds that, for $e \sim \mathcal{N}(0,\sigma^2)$ and $t > 0$:

$$\Pr[e > t] = \frac{1}{2}\,\mathrm{erfc}\left(\frac{t}{\sqrt{2}\sigma}\right) \ .$$

We let $i = a..b$ denote an iterative integer index i that ranges from a to b, and let $[a]$ denote the set $\{0, 1, \cdots, a-1\}$ for any integer a. We let $\mathrm{Avg}(\mathbf{a})$ denote an average of the coefficients of a vector $\mathbf{a}$.

2.1 Fully Homomorphic Encryption

A fully homomorphic encryption scheme is an encryption scheme that enables the evaluation of circuits on the data underlying ciphertexts.

Definition 2.1 (Fully homomorphic encryption)**.** A fully homomorphic encryption scheme (FHE) is a tuple of efficient algorithms (KeyGen, Enc, Dec, Eval) with the following specifications:

- KeyGen outputs a secret key sk and a public key pk;
- Enc takes as inputs a public key pk and a plaintext $m \in \{0,1\}$, and outputs a ciphertext ct;
- Eval takes as inputs a public key pk, a binary circuit C and a tuple of ciphertexts $\mathrm{ct}_1, \ldots, \mathrm{ct}_k$ where k is the number of input wires of C, and outputs a ciphertext ct;
- Dec takes as inputs a secret key sk and a ciphertext ct, and outputs a plaintext m.

For $\varepsilon \geq 0$ and a circuit size bound B, the scheme is said (ε, B)-correct if for any pair (sk, pk) output by KeyGen, for any binary circuit C of size at most B, for any plaintexts $m_1, \ldots, m_k \in \{0,1\}$ where k is the number of input wires of C, the following holds with probability $\geq 1-\varepsilon$ over $\mathrm{ct}_i := \mathsf{Enc}(\mathsf{pk}, m_i)$ (for $i \leq k$):

$$\mathsf{Dec}_{\mathsf{sk}}\left(\mathsf{Eval}_{\mathsf{pk}}(C, (\mathrm{ct}_1, \ldots, \mathrm{ct}_k))\right) = C(m_1, \ldots, m_k) \ .$$

An FHE scheme is perfectly correct if it is ε-correct for $\varepsilon = 0$ and any B.

Typical FHE definitions also include a ciphertext compactness condition, to avoid vacuous constructions. We omit it as this is irrelevant to our work. We note that the plaintext space of concrete FHEs can be more complex than $\{0,1\}$, such as rings $\mathbb{Z}/k\mathbb{Z}$ for some integer k or Cartesian products of such rings, or approximations to $\mathbb{R}^d$ or $\mathbb{C}^d$. In the latter case, the FHE is said approximate. When it enables exact computations on discrete data, it is said exact.

Concerning correctness, we note that most concrete schemes are not ε-correct as per the above definition. All known FHE constructions have a notion of noise, which goes throughout homomorphic manipulations. The noise must remain sufficiently small to enable correct decryption. To obtain better parameters, concrete schemes often make heuristic assumptions on noise growth, notably relying on probabilistic arguments without any randomness. To decrease the noise, the known constructions rely on an operation called bootstrapping, which has a (heuristic) failure probability. For very long computations, bootstrapping may be required many times, so that ε-correctness cannot be ensured, for any $\varepsilon < 1$.

The default notion of security for an FHE is indistinguishability against chosen plaintext attacks (IND-CPA).

Definition 2.2 (IND-CPA security game)**.** For an FHE scheme Π = (KeyGen, Enc, Dec, Eval), we define the advantage $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}(\mathcal{A})$ of an adversary $\mathcal{A}$ against the IND-CPA security of Π as:

$$\left| 2 \cdot \Pr\left[b = b' \;\middle|\; \begin{array}{c} (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}; \\ b \leftarrow \mathcal{U}(\{0,1\}); \\ b' \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{Enc}(b)) \end{array} \right] - 1 \right| \ .$$

In the context of FHE, IND-CPA security is typically inherited from the presumed hardness of LWE [43], RLWE [39, 44], GLWE [15, 35] and circular security assumptions.

2.2 IND-CPA^D Security

IND-CPA^D security, introduced in [37], augments IND-CPA security by allowing the adversary to access decryptions of ciphertexts obtained by encrypting and evaluating messages of its choice. This is defined formally by allowing the adversary to access three types of oracles, to encrypt, evaluate and decrypt. Decryption queries can be made only on ciphertexts produced by the encrypt/evaluate oracle. This is simply to ensure that these ciphertexts are well-formed. The database S appearing in the oracles is used to store all ciphertexts formed by using the encryption and evaluation oracles as well as the underlying pair of plaintexts (m_0, m_1), where m_b is the encrypted plaintext (possibly the result of some prior evaluations) for b being the challenge bit. Notably, the database is used to ensure that a decryption query made on a ciphertext does not lead to a trivial attack by checking that the underlying plaintext is independent of the challenge bit b.

Algorithm 1 Encryption oracle $O_{\mathsf{Enc}}(m_0, m_1; \mathsf{pk}, b, i)$

$\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_b)$
$S[i] := (m_0, m_1, \mathsf{ct})$
$i := i + 1$
return ct, i

Algorithm 2 Evaluation oracle $O_{\mathsf{eval}}(C, i_1, \ldots, i_k; b, S)$

$\mathsf{ct} \leftarrow \mathsf{Eval}_{\mathsf{pk}}(C, S[i_1].\mathsf{ct}, \ldots, S[i_k].\mathsf{ct})$
$r_0 := C(S[i_1].m_0, \ldots, S[i_k].m_0)$
$r_1 := C(S[i_1].m_1, \ldots, S[i_k].m_1)$
$S[i] := (r_0, r_1, \mathsf{ct})$
$i := i + 1$
return ct, i

Algorithm 3 Decryption oracle $O_{\mathsf{Dec}}(j; \mathsf{sk}, S)$

if $S[j].m_0 = S[j].m_1$ **then**
  $m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(S[j].\mathsf{ct})$
  return m
else
  return $\perp$
end if

Definition 2.3 (IND-CPA^D security)**.** Let Π = (KeyGen, Enc, Dec, Eval) be an FHE scheme. For an adversary $\mathcal{A}$ that is given access to the (stateful) oracles O_{Enc}, O_{Eval} and O_{Dec} defined above, we define the advantage $\mathsf{Adv}^{\mathsf{INDCPAD}}(\mathcal{A})$ of $\mathcal{A}$ against the IND-CPA^D security of Π as:

$$\left| 2 \cdot \Pr\left[b = b' \;\middle|\; \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}; \\ b \leftarrow \mathcal{U}(\{0,1\}); \\ b' \leftarrow \mathcal{A}^{O_{\mathsf{Enc}}, O_{\mathsf{Eval}}, O_{\mathsf{Dec}}}(\mathsf{pk}) \end{array} \right] - 1 \right| .$$

2.3 KR^D Security

KR^D security is a relaxation of IND-CPA^D security introduced in [38]. Given the same types of oracles, the adversary's task is to recover the secret key, as opposed to distinguishing between two (families of) ciphertexts.

Definition 2.4 (KR^D security)**.** Let Π = (KeyGen, Enc, Dec, Eval) be an FHE scheme. For an adversary $\mathcal{A}$ that is given access to the (stateful) oracles O_{Enc}, O_{Eval} and O_{Dec} with b fixed to 0, we define the success probability $\mathsf{Succ}^{\mathsf{KRD}}(\mathcal{A})$ against the KR^D security of Π as:

$$\left| \Pr\left[\mathsf{sk} = \mathsf{sk}' \;\middle|\; \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}; b = 0; \\ \mathsf{sk}' \leftarrow \mathcal{A}^{O_{\mathsf{Enc}}, O_{\mathsf{Eval}}, O_{\mathsf{Dec}}}(\mathsf{pk}, b) \end{array} \right] \right| .$$

Then, an FHE scheme is said KR^D secure if the success probability of any PPT adversary $\mathcal{A}$ is negligible on λ.

If a scheme is IND-CPA^D secure, then it is KR^D secure: indeed, any adversary $\mathcal{A}$ against KR^D security of Π leads to an adversary $\mathcal{B}$ against IND-CPA^D security of Π with the same advantage and the same runtime. As a result, an attack against KR^D security is stronger than an attack against IND-CPA^D security.

3 KR^D Attacks on B(F/G)V-like FHE Schemes

In this section, we present our KR^D attack against the BGV and BFV (F)HE schemes [14, 15, 27] and their variants, when they rely on (heuristically) average-case noise analysis. We note that our attack also holds for the leveled version of those schemes (i.e., without bootstrapping), with no level consumption. For the sake of simplicity, we focus on the BFV scheme, and its implementation in OpenFHE [7]. We stress that our attack also applies to other implementations of BGV and BFV that rely on average-case noise analyses. For instance, HElib uses a worst-case noise analysis after encryption, thus our attack is not applicable. However, many recent instantiations of BGV and BFV [7, 9, 42] rely on average-case noise analyses to improve performance. These are vulnerable to our attack.

3.1 Noise Analysis for BFV

We start with the following notations for the BFV FHE scheme.

- $N > 0$: the ring degree;
- $\mathcal{R}$: the base ring, of the form $\mathbb{Z}[X]/\Phi(X)$ for some cyclotomic polynomial of degree N;
- $q > 0$: the ciphertext modulus;
- $\mathcal{R}_q$: the quotient ring $\mathcal{R}/q\mathcal{R}$;
- $t > 0$: the plaintext modulus;
- $\Delta = \lfloor q/t \rceil$: the scale factor.

The RLWE key $\mathbf{s} \in \{-1, 0, 1\}^N \subset \mathcal{R}$ is chosen ternary, possibly sparse (i.e., with many 0's).

Let us start with a RLWE-format ciphertext $\mathrm{ct} = (\mathbf{b}, \mathbf{a}) \in \mathcal{R}_q^2$ satisfying

$$\mathbf{b} + \mathbf{as} = \Delta\mathbf{m} + \mathbf{e} \bmod q \ ,$$

for some plaintext $\mathbf{m} \in \mathcal{R}_t$ and some error $\mathbf{e} \in \mathcal{R}$. For a fresh ciphertext, the coefficients of $\mathbf{e}$ are independently sampled from D_σ, a discrete Gaussian distribution over $\mathbb{Z}$ centered in 0 with standard deviation σ. For two ciphertexts, we define their homomorphic addition as their componentwise addition in $\mathcal{R}_q^2$, i.e., for $\mathrm{ct}_0 = (\mathbf{b}_0, \mathbf{a}_0)$ and $\mathrm{ct}_1 = (\mathbf{b}_1, \mathbf{a}_1)$ with underlying errors $e_0, e_1 \leftarrow D_\sigma$, their homomorphic addition is

$$(\mathbf{b}_0 + \mathbf{b}_1 \bmod q, \mathbf{a}_0 + \mathbf{a}_1 \bmod q) \ ,$$

resulting in an underlying error term that is equal to $\mathbf{e}_0 + \mathbf{e}_1$. This error has coefficients that have a variance of $\approx 2\sigma^2$ if the errors $\mathbf{e}_0$ and $\mathbf{e}_1$ are sampled independently and identically from D_σ. However, if the two ciphertexts are correlated, the variance can be smaller, or larger. In the worst case, we have $\mathrm{ct}_0 = \mathrm{ct}_1$ and the error term $2\mathbf{e}_0$ has a variance of $4\sigma^2$. The most efficient instantiations only consider the first option, as this leads to improved parameters and performance. However, it may result in underestimating the error probability.

The evaluation (or the decryption of the evaluated result) is allowed if the variance is sufficiently small compared to the threshold $\Delta/2$. More precisely, the probability of evaluation failure is bounded from above by $\mathrm{erfc}((\Delta/2)/(\sqrt{2}\sigma))$, where σ is the standard deviation of the error in the ciphertext under scope.

3.2 Key-Recovery Attack

As discussed above, the average-case noise analysis does not capture the different types of variance increases, depending on whether the ciphertexts are statistically independent or correlated. Our attack takes advantages of this gap, which can be exploited by the attacker, and the ciphertext can be decrypted to a message other than what it should have been, in the view of the challenger.

Algorithm 4 KR$^{\mathsf{D}}$ attack on BFV.

1: Query $\mathrm{ct}_0 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(0)$
2: **for** $0 \le i < k := \lfloor \log_2 \Delta \rceil$ **do**
3: Query $\mathrm{ct}_{i+1} \leftarrow \mathsf{Eval}_{\mathsf{pk}}(\mathsf{Add}, \mathrm{ct}_i, \mathrm{ct}_i)$
4: **end for**
5: Query $\mathbf{e} \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}_k)$
6: Solve $\mathbf{b} - \mathbf{e} = \mathbf{as}$ over $\mathcal{R}_q$
7: **return s**

The attack is given in Algorithm 4. Note that the calls to Enc, Eval and Dec correspond to oracle queries made to the challenger, as per Definition 2.4. The attack is pictorially represented in Figure 1. For the sake of simplicity, we focus on the power-of-2 integer for scale factor Δ.

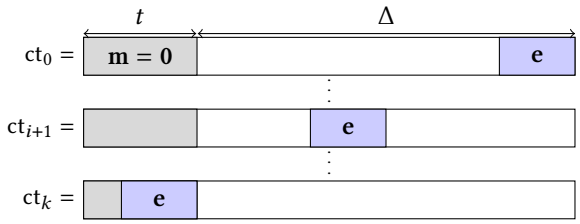


Figure 1. Position of the error in $\mathbf{b} + \mathbf{as} \bmod q$ during the iterations of the KR$^{\mathsf{D}}$ attack.

Our attack proceeds as follows. We first query an encryption of 0 and obtain a ciphertext ct_0, whose error $\mathbf{e}$ is sampled from D_σ. We then query evaluations of additions, recursively to itself. After $k = \lceil \log_2 \Delta \rceil$ iterations, we query a decryption oracle to the ciphertext. From the decryption result, we recover the initial error $\mathbf{e}$, then recover the secret key by solving a linear equation.

During the iterative additions, the standard deviation of the resulting error is doubled for each iteration. After k iterations, the error has a standard deviation $2^k\sigma \approx \Delta\sigma$. However, the average-case analysis estimates that the error standard deviation is $2^{k/2}\sigma \approx \sqrt{\Delta}\sigma$. Thus, the decryption failure probability will be estimated to $\approx \mathrm{erfc}((\Delta/2)/(\sqrt{2\Delta}\sigma))$ based on the average-case analysis. Unless this is deemed too large, the decryption oracle is allowed. The attacker receives the decryption result

$$\left\lfloor \frac{2^k\mathbf{e} \bmod q}{\Delta} \right\rceil = \mathbf{e} \ ,$$

unless the infinite norm of $\mathbf{e}$ is larger than t. If the error $\mathbf{e}$ has infinite norm larger than t, we may repeat the process with $k-1$, $k-2$, etc iterations instead of k, to learn $\mathbf{e}$, chunk by chunk. The attacker can then recover the secret key by solving a linear equation $\mathbf{as} = \mathbf{b} - \mathbf{e}$ over $\mathcal{R}_q$ with unknown $\mathbf{s}$. If it needs a few more equations to find the solution (note that $\mathcal{R}_q$ is not a field), it may repeat the process several times.

For a general Δ, we can learn $\mathbf{e}$ from its scaled and rounded values, repeating the process.

3.3 Experimental Results

For our experiments, we consider a typical BFV parameter set generated by the OpenFHE library. We have $N = 4096$, $q = 2^{60}$, $t = 2^{16} + 1$, and $\Delta = 2^{44} - \varepsilon$ for some small ε. We use uniform ternary secret. The standard deviation of the initial error is $\approx 2^{7.41}$. We start from a fresh encryption of 0 that we recursively double $k = 44$ times. The decryption failure probability estimate based on the average-case approach is $\approx \mathrm{erfc}(2^{13.5}) \approx 2^{-2^{27.5}}$, which is extremely low. Thus the decryption is allowed and the decryption result is

$$\left\lfloor \frac{2^{44}\mathbf{e}}{2^{44} - \varepsilon} \right\rceil = \mathbf{e} + \left\lfloor \frac{\varepsilon\mathbf{e}}{2^{44} - \varepsilon} \right\rceil = \mathbf{e} \ ,$$

unless the infinite norm of $\mathbf{e}$ is larger than $(2^{44}-\varepsilon)/(2\varepsilon) \approx 2^{16}$. This is very unlikely to happen, since each coefficient has a standard deviation of $\approx 2^7$.

With the settings above, we perfectly recovered the error after decryption. Now, we recover the secret key by solving the linear equation $\mathbf{as} = \mathbf{b} - \mathbf{e}$ over $\mathcal{R}_q$ in several seconds, as in [37]. We first compute the inverse of $\mathbf{a}$ over the ring, then multipy it to $\mathbf{b} - \mathbf{e}$, which will recover the full secret key.

3.4 On the Default BFV Implementation of OpenFHE

The OpenFHE implementation of BFV does not rely on the agressive error analysis we considered so far, based on the heuristic assumption that the noise terms of ciphertexts to be added would be statistically independent. It instead uses the triangular inequality to bound the error growth for additions. In particular, as mentioned in [7, Section 2.6.1] and recalled in [5, Section 5.2], OpenFHE provides an option to set the maximum allowed number of additions per multiplication level, at parameter generation.[2] The corresponding variable is SetEvalAddCount, and the error due to additions is bounded from above as the input error bound, multiplied by SetEvalAddCount.

The above approach misses the fact that the result of an addition can itself be an input to a subsequent addition, leading to a larger noise amplification. Indeed, if we repeat k times the addition of a number with itself, only k additions are performed, but the noise grows as 2^k. This is exactly what Algorithm 4 does. As a result, it also provides an attack against the default OpenFHE implementation of BFV, even when used as intended. Instead of exploiting a heuristic aspect of the noise analysis, it exploits an incorrect worst-case noise analysis.

4 KR^{D} Attacks from Too Large Bootstrapping Failure Probability

In this section, we present our KR^{D} attack against the DM and CGGI FHE schemes [19, 26], also respectively known as FHEW and TFHE, and discrete FHE schemes based on CKKS [3, 20, 25, 31]. Interestingly, our attack also applies to the CKKS scheme *over approximate numbers* with noise flooding countermeasures [38], when the bootstrapping failure probability is not sufficiently low. The KR^{D} attacker can observe decryption failures and obtain information from the failing events. Our KR^{D} attack takes advantage of the errors introduced during bootstrappings, that give *approximate inequality hints* on the secret key when the bootstrapping fails.

In the following, we first recall the approximate inequality hints, and introduce a simple yet powerful method to utilize these hints to recover the secret key. We then introduce how the inequality hints can be obtained from the bootstrappings of DM/CGGI and CKKS. Finally, we illustrate our attacks with experiments.

[2] See for instance, 'ParameterGenerationBFVRNS::ParamsGenBFVRNS' function of [7] which generates parameters based on a noise analysis regarding the number of allowed additions per level.

4.1 Inequality Hints

We first define the approximate inequality hint on a secret vector in Definition 4.1, which is a type of approximate hint [21].

Definition 4.1 (Approximate Inequality Hint)**.** For $\sigma > 0$ and a threshold $t > 0$, a (σ, t)-approximate-inequality hint on the secret vector $\mathbf{z} \in \{0, \pm 1\}^N$ is a vector $\mathbf{c} \in \mathbb{R}^N$ such that

$$\langle \mathbf{c}, \mathbf{z} \rangle + e > t \ ,$$

where $e \sim \mathcal{N}(0, \sigma^2)$.

In our attack scenario, the hint vectors are originated from a uniform distribution on an interval centered in 0. For the sake of simplicity, let us assume that $\mathbf{c} \sim \mathcal{U}([-1/2, 1/2]^N)$ and $\mathbf{z} \in \{0, 1\}^N$. It is not surprising that if $\mathbf{c}$ is a hint vector on a secret vector $\mathbf{z}$, then the hints and the secret are aligned: the larger the threshold t, the more aligned they are. This leads us to the following statement, on the distribution of a vector $\mathbf{c}$, conditioned on whether it is a hint on the secret vector $\mathbf{z}$ or not.

Lemma 4.2. *Let $\sigma > 0$, $t > 0$ and a secret vector $\mathbf{z} \in \{0, 1\}^N$. Let $\mathbf{c} \sim \mathcal{U}([-1/2, 1/2]^N)$ and $e \sim \mathcal{N}(0, \sigma^2)$. Let $i \in [N]$ and $Y_i = \langle \mathbf{c}, \mathbf{z} \rangle - c_i z_i$. Then the conditional probability density function f of c_i conditioned on the event $\langle \mathbf{c}, \mathbf{z} \rangle + e > t$ satisfies, for all $x_i \in [-1/2, 1/2]$:*

$$f(x_i) = \begin{cases} \frac{\Pr[x_i + Y_i + e > t]}{\Pr[c_i + Y_i + e > t]} & \text{if } z_i = 1, \\ 1 & \text{if } z_i = 0. \end{cases}$$

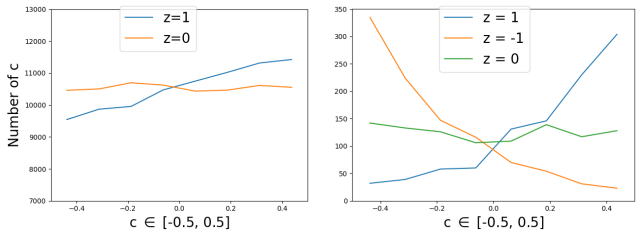
Proof. We consider two cases. If $z_i = 0$, the failure event is independent of the random variable c_i and its conditional distribution remains the uniform distribution on $[-1/2, 1/2]$. If $z_i = 1$, Bayes' law gives

$$f(x_i) = \frac{\Pr[\langle \mathbf{c}, \mathbf{z} \rangle + e > t \mid c_i = x_i]}{\Pr[\langle \mathbf{c}, \mathbf{z} \rangle + e > t]} \ .$$

We use the equality $c_i z_i + Y_i = \langle \mathbf{c}, \mathbf{z} \rangle$ to conclude. □

Lemma 4.2 implies that the distribution of the coefficient c_i is very different depending on the value $z_i \in \{0, 1\}$. Indeed, one is uniform while the other has an increasing density function (if $t > 0.5$). Since Y_i is a sum of $(\mathrm{hw}(\mathbf{z}) - 1)$ uniform variables (when $z_i = 1$), if the Hamming weight of $\mathbf{z}$ is sufficiently large, $Y_i + e$ approximately follows $\mathcal{N}(0, \sigma^2 + (\mathrm{hw}(\mathbf{z}) - 1)/12)$. Thus the density function for $z_i = 1$ is similar to a scaled and reversed erfc function at $> t - x_i$.

In our attacks, the source of the hints is the bootstrapping failures, and in practice, t is chosen so that the bootstrapping failure probability is not too high. Thus, we are indeed somewhat far from the zero in the erfc function, in a range



(a) CGGI **(b)** CKKS

Figure 2. Distributions of coefficients c_i of hints obtained from (a) CGGI bootstrapping for $z_i \in \{0, 1\}$, and (b) CKKS bootstrapping for $z_i \in \{0, \pm 1\}$. Custom parameters with 128-bit IND-CPA security are used (see Section 4.4 for details). The x axis shows the range of c_i and the y axis shows the number of c_i's in each interval of length $1/8$.

where it varies significantly. Figure 2 displays the distribution of c_i of the approximate inequality hints obtained from the custom parameters for CGGI and CKKS implemented in TFHE-rs and Lattigo, respectively. We note that we can similarly exploit the distributions of the c_i's conditioned on correct bootstrappings, but then the two distributions are extremely close.

Distinguishing the two distributions. Our attacks exploit the different tendencies of the probability density functions. For instance, the average of the coefficients c_i over the different hint vectors $\mathbf{c}$ will give sufficient information about z_i. The average corresponding to $z_i = 0$ is 0, whereas one corresponding to $z_i = 1$ is a positive value. We can categorize the indices based on the average, depending on whether it is larger than a threshold, or not. It is worth noting that the slope of the hints distribution in relation to $z_i = 1$ has an impact on both the quality and cost of the attack: the steeper the slope, the easier it is to distinguish between the two distributions.

More concretely, assuming the independence of the failing ciphertexts, the average $\tilde{\mathbf{c}} = \mathrm{Avg}(\mathbf{c}) \in \mathbb{R}^N$ over the f hints is a sum of f i.i.d. random variables. If $z_i = 0$, $\tilde{c}_i$ is a scaled sum of f uniform variables, and it follows the Irwin-Hall distribution. It can be approximated to Gaussian distribution centered at 0 with variance $\sigma_0^2 = 1/(12f)$, using the central limit theorem when f is sufficiently large. If $z_i = 1$, $\tilde{c}_i$ is a scaled sum of f random variables, each following an identical distribution of variance $\sigma_{\mathsf{trunc}}^2 \ll \sigma^2 + (\mathrm{hw}(\mathbf{z}) - 1)/12$. Using the central limit theorem, we can approximate the distribution of $\tilde{c}_i$ to Gaussian distribution centered at some $\alpha > 0$ with variance $\sigma_1^2 = \sigma_{\mathsf{trunc}}^2/f$. Therefore, when using a threshold of $\alpha/2$, the probability of correctly categorizing c_i will be a linear combination of two erfc values, for inputs $(\alpha/2)/(\sqrt{2}\sigma_0)$ and $(\alpha/2)/(\sqrt{2}\sigma_1)$. To correctly categorize all $N \lesssim 2^{16}$ coefficients, it is sufficient to have the inputs to erfc functions larger than 3. This leads us the following claim.

Theorem 4.3 (informal). *Given (σ, t)-approximate-inequality hints $c_0, \cdots, c_{f-1} \in \mathbb{R}^N$ on a secret vector $\mathbf{z} \in \{0, 1\}^N$, we can recover the secret vector with probability $O(1)$, if $N \leq 2^{16}$ and f satisfies $f \geq \max(3/(2\alpha^2), (3\sigma_{\mathsf{trunc}}/\alpha)^2)$, where σ_{trunc} is the standard deviation of the Irwin-Hall distribution for $(\mathrm{hw}(\mathbf{z}) - 1)$ sums of $\mathcal{U}([-1/2, 1/2])$, truncated at t.*

In practice, we can approximate the value $\alpha/2$ by averaging $\tilde{\mathbf{c}}$ over $z_i \in \{0, 1\}$. In the case when the Hamming weight of the secret is given, we can directly choose the largest $\mathrm{hw}(\mathbf{z})$ values from $\tilde{\mathbf{c}}$.

We note that, if t gets larger, the failure probability decreases, however, the standard deviation σ_{trunc} of the truncated distribution gets smaller and α becomes larger. Thus, fewer hints are required to recover the full secret, even if the failure probability is lower than before. For the ternary secret, we can simply extend all the results from above.

4.2 KR^D Attacks on DM/CGGI

For the sake of simplicity, we only focus on the CGGI scheme and the TFHE-rs library [47], but stress that our attack is general and not specific to our choice of variant for DM/CGGI and our choice of target implementation. Our KR^D attack against DM/CGGI FHE schemes takes advantage of the large rounding error in the ModSwitch step during gate bootstrapping.

In the following, we first recall the CGGI (gate) bootstrapping, then introduce our KR^D attack.

TFHE (Gate) Bootstrapping. We use the following notations for the CGGI scheme parameters.

- $n > 0$: the LWE dimension;
- $q > 0$: the initial modulus of ciphertexts;
- N: the ring degree, set as a power-of-2 integer;
- $p > 0$: the plaintext modulus;
- $\Delta = q/p$: the scale factor.

The LWE secret key $\mathbf{s} = (s_1, \ldots, s_n) \in \{0, 1\}^n$ is chosen binary.

TFHE bootstrapping. Let us start with an LWE ciphertext $\mathrm{ct} = (b, a_1, \ldots, a_n) \in \mathbb{Z}_q^{n+1}$ satisfying

$$-b + \sum_{i=1}^{n} a_i s_i = \Delta m + e_{\mathsf{in}} \bmod q \ ,$$

for some $m \in \{0, \ldots, p-1\}$ and some integer e_{in} whose absolute value is small relative to Δ. The bootstrapping procedure consists of 4 steps, namely, ModSwitch, BlindRotate, SampleExtract, and KeySwitch. An illustration of bootstrapping is provided in Figure 3.

Note that the error (relatively to the modulus) is decreased by BlindRotate, and is unchanged or increased at all other steps. We will focus on ModSwitch because 1) the error incurred by this step is quite large and 2) information concerning this error is publicly available.

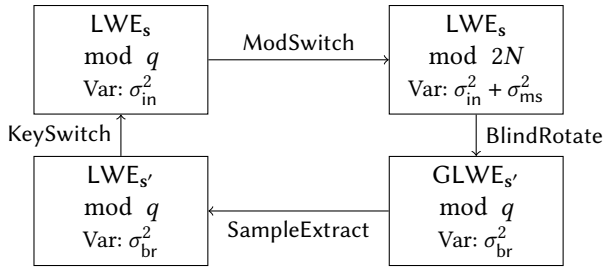


Figure 3. High-level overview of the TFHE bootstrapping loop. In each box, we give the ciphertext format, the secret key, the current ciphertext modulus, and the (heuristic) normalized variance of the noise after decryption with the secret key (the variance is normalized by the square of the modulus). The initial variance σ^2_{in} can vary depending on the input: a fresh ciphertext (σ^2_{fresh}), a bootstrapped ciphertext ($\sigma^2_{\text{br}} + \sigma^2_{\text{ks}}$), or a linear combination of them (for gate bootstrapping).

ModSwitch maps a ciphertext $\text{ct} = (b, a_1, \cdots, a_n) \in \mathbb{Z}_q^{n+1}$ with modulus q to a ciphertext $\tilde{\text{ct}} = (\tilde{b}, \tilde{a}_1, \cdots, \tilde{a}_n) \in \mathbb{Z}_{2N}^{n+1}$ with modulus $2N$ with $q \gg 2N$, by rounding:

$$\tilde{b} = \left\lfloor \frac{2N \cdot b}{q} \right\rceil,\ \tilde{a}_i = \left\lfloor \frac{2N \cdot a_i}{q} \right\rceil, \forall i \leq n \ .$$

For each coefficient, this rounding creates a rounding error $\tilde{e}_i$ which is a deterministic function of the publicly available ciphertext:

$$\tilde{e}_0 = \left\lfloor \frac{2N \cdot b}{q} \right\rceil - \frac{2N \cdot b}{q},\ \tilde{e}_i = \left\lfloor \frac{2N \cdot a_i}{q} \right\rceil - \frac{2N \cdot a_i}{q} \ ,$$

for $i \leq n$. We have

$$-\tilde{b} + \sum_{i=0}^{n-1} \tilde{a}_i s_i = \frac{2N}{q}(\Delta m + e_{\text{in}}) + \langle \tilde{\mathbf{e}}, (-1, \mathbf{s}) \rangle \bmod 2N \ . \quad (4.1)$$

We have a pre-ModSwitch error e_{in} with standard deviation σ_{in} and a new error $e_{\text{ms}} = \langle \tilde{\mathbf{e}}, (-1, \mathbf{s}) \rangle$ with standard deviation σ_{ms}. Assuming that $\tilde{\mathbf{e}} \sim U\left((-1/2, 1/2]^{n+1}\right)$ and that $\mathbf{s}$ has as many 1's as 0's, the normalized standard deviation of e_{ms} is

$$\sigma_{\text{ms}} = \sqrt{\frac{1}{12}\left(1 + \frac{n}{2}\right)} \cdot \frac{1}{2N} \ .$$

We note that this bound assumes that the Hamming weight of the secret key vector is $n/2$, which is the most likely situation but may not be the case for specific secret keys.

We emphasize that the rounding error $\tilde{\mathbf{e}}$ is publicly computable. We also argue that it is typically large relatively to the new modulus $2N$: it is (heuristically) uniform in $(-1/2, 1/2]$; and N is set as small as possible as it has a strong impact on the bootstrapping performance.

Correctness of bootstrapping. Assume we start with a ciphertext modulo q that decrypts to a plaintext m under key $\mathbf{s}$. We say that bootstrapping fails if after ModSwitch, BlindRotate, SampleExtract, and KeySwitch, the new ciphertext does not decrypt to m anymore. Recall that decryption fails (i.e., the decryption result is incorrect) if the error in the ciphertext exceeds a threshold. As the noise keeps increasing (except with BlindRotate), the moment when it is most likely to provide a decryption error is after ModSwitch. Several errors contribute: those introduced during BlindRotate, those introduced by KeySwitch, and those introduced by ModSwitch. In practice, the largest one is typically the one introduced by ModSwitch.

Before ModSwitch, the ciphertext is associated to an error e_{in} of (heuristic, normalized) standard deviation σ_{in} satisfying $\sigma^2_{\text{in}} = \sigma^2_{\text{br}} + \sigma^2_{\text{ks}}$, where the term σ_{ks} corresponds to the error introduced by KeySwitch. ModSwitch introduces an extra error term e_{ms}, whose normalized standard deviation is σ_{ms}. Note that we heuristically assume all errors from all steps to be statistically independent. The total relative error after ModSwitch is given as $(e_{\text{in}}/q + e_{\text{ms}})/(2N)$, which has variance $\sigma^2_{\text{bts}} = \sigma^2_{\text{br}} + \sigma^2_{\text{ks}} + \sigma^2_{\text{ms}}$. Assuming that the error behaves as a continuous Gaussian, the probability of incorrect bootstrapping is computed as:

$$\text{erfc}\left(\frac{1/16}{\sqrt{2} \cdot \sigma_{\text{bts}}}\right) \ .$$

Here $1/16$ is the relative threshold for the correct decryption: when the error is above this value, the plaintext obtained by decrypting after bootstrapping may differ from the initial plaintext m.

Correctness of gate bootstrapping. Gate bootstrapping combines the evaluation of a binary gate and bootstrapping. This is achieved by adding two ciphertexts and a constant before, or after KeySwitch, and running bootstrapping as above. When to add the two ciphertexts depends on the parameter choice. We consider the case of adding the ciphertexts after KeySwitch, as this corresponds to the default situation in the TFHE-rs library, but we stress that the attack works for both cases (with success probabilities depending on the parameter choices).

The addition of the two ciphertexts corresponds to an addition of the underlying plaintexts over the integers, which suffices for gate evaluation as any symmetric binary gate is a function of the integer sum of the input bits. The ciphertext ct then goes through ModSwitch, BlindRotate, etc. If $\text{ct}_0 = \text{ct}_1$, we have $\sigma^2_{\text{in}} = 4(\sigma^2_{\text{br}} + \sigma^2_{\text{ks}})$ and thus $\sigma^2_{\text{gbts}} = 4\sigma^2_{\text{br}} + 4\sigma^2_{\text{ks}} + \sigma^2_{\text{ms}}$ after ModSwitch. Assuming that the error behaves as a continuous Gaussian, the probability of incorrect bootstrapping is $\text{erfc}(1/(16\sqrt{2} \cdot \sigma_{\text{gbts}}))$.

Inequality hints from CGGI gate bootstrapping failures. In the KR^{D} security model, the attacker has accesses to

a decryption oracle (limited to properly created ciphertexts). Decryption takes place for LWE ciphertexts under modulo q under key $\mathbf{s}$, i.e., after KeySwitch and before ModSwitch. Decryption after (gate) bootstrapping can fail for several reasons:

- post-ModSwitch error $e_{\mathsf{in}} + e_{\mathsf{ms}}$ is too large;
- pre-decryption (post-KeySwitch) error $e_{\mathsf{br}} + e_{\mathsf{ks}}$ is too large.

As the purpose of bootstrapping is reducing the error, the parameters are set to have lower (relative) standard deviation for the second types of error than the first type. This means, the relative standard deviation of the error reaches the maximum after ModSwitch, and if the (gate) bootstrapping fails, it is extremely likely that the error after ModSwitch was above the correct decryption threshold. This induces a (two-sided) approximate inequality hint $\tilde{\mathbf{e}}$ for the secret vector $(-1, \mathbf{s})$, where $\tilde{\mathbf{e}}/2N \sim \mathcal{U}((-1/2, 1/2]^{n+1})$, and error e_{in}/q with standard deviation of σ_{in}.

To obtain a one-sided hint, we focus on the gate bootstrapping for AND gates. Concretely in TFHE-rs, the AND gate is implemented by first computing $\mathsf{ct} = \mathsf{ct}_0 + \mathsf{ct}_1 - (q/8, 0, \cdots, 0)$ then applying the bootstrapping on ct, for the input ciphertexts ct_0 and ct_1. Noting that, $q/8$ and $3q/8$ correspond to true whereas $-3q/8$ and $-q/8$ correspond to false, the gate bootstrapping failure with input ciphertexts both encrypting true implies a one-sided hint. Concretely from Equation 4.1, we have

$$\langle \tilde{\mathbf{e}}, (-1, \mathbf{s}) \rangle + \frac{2N}{q} e_{\mathsf{in}} > \frac{2N}{16} \quad .$$

Algorithm 5 KR$^{\mathsf{D}}$ attack on TFHE

$f := 0$
for $0 \le k < \gamma$ **do**
  Query $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{true})$
  Query $\mathsf{ct}' \leftarrow \mathsf{Eval}(\mathsf{AND}, \mathsf{ct}, \mathsf{ct})$
  Query $\mathsf{ct}'' \leftarrow \mathsf{Eval}(\mathsf{AND}, \mathsf{ct}', \mathsf{ct}')$
  Query $m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}'')$
  if $m = \mathsf{false}$ **then**
    Compute $\tilde{\mathbf{e}} \leftarrow \mathsf{GenModSwitchError}(\mathsf{ct}')$, then let $\mathbf{c}_f = \tilde{\mathbf{e}}/2N$
    $f := f + 1$
  end if
end for
Compute $\tilde{c}_i = \mathrm{Avg}_{j \in [f]}(c_{j,i})$ for $i \in [n+1]$
Compute $s_{\mathsf{est},i} = \begin{cases} 1 & \text{if } \tilde{c}_i > \alpha/2, \\ 0 & \text{otherwise,} \end{cases}$ for $i \in [n+1]$
return $\mathbf{s}_{\mathsf{est}}$

KR$^{\mathsf{D}}$ **attack against CGGI.** Our KR$^{\mathsf{D}}$ attack is given in Algorithm 5, which proceeds as follow. We first query an encryption of true, then we query an AND gate twice on the resulting ciphertext itself. Then we ask for a decryption. When failure occurs, an approximate inequality hint is obtained. We collect such hints, then compute the average per each index. Finally, we categorize the indices key based on the average, and output $\mathbf{s}_{\mathsf{est}}$, a guess for the secret key $\mathbf{s}$. The calls to Enc, Eval and Dec formally correspond to oracles to the challenger, as per Definition 2.4. The parameter γ quantifies the number of attempts to create decryption failures, and the parameter $\alpha/2$ is a threshold for categorizing. Note GenModSwitchError denote the procedure generating the rounding error $\tilde{\mathbf{e}}$ from ct′, which do not require the secret key.

The first bootstrapping increases the variance of the error of the ciphertext, which will be input to the AND gate bootstrapping. This accelerates collecting the gate bootstrapping failures. The average of the hints includes information about the distributions of $\tilde{e}$, and thus of $\mathbf{s}$. The attack recovers the full secret key if f is sufficiently large, thanks to Theorem 4.3. The threshold, in practice, can be chosen the average of the averages, i.e., $\alpha/2 = \mathrm{Avg}(\tilde{\mathbf{e}})$.

4.3 KR$^{\mathsf{D}}$ Attacks on CKKS over discrete data

Recent works introduce techniques for discrete computations using CKKS instead of BGV/BFV by discretizing the message space [3, 20, 25]. Another recent work proposes to use the CKKS bootstrapping as a subroutine to accelerate the expensive BGV/BFV bootstrappings [31]. In all of the above cases, to achieve the IND-CPA$^{\mathsf{D}}$/KR$^{\mathsf{D}}$ security, CKKS should be correct within a given threshold, with overwhelming probability. However, the CKKS bootstrapping is not perfectly correct, and if it fails, the adversary obtains inequality hints about the secret key. This type of bootstrapping failure also occurs in the CKKS variant introducing noise flooding countermeasures [38] if one does not set the bootstrapping failure probability to be low enough [7].

In the following, we first recall the CKKS bootstrapping, then introduce our KR$^{\mathsf{D}}$ attack strategy.

CKKS bootstrapping. We start with the following notations for the CKKS FHE scheme.

- $N > 0$: the ring degree, power-of-two integer;
- $\mathcal{R}$: the base ring, of the form $\mathbb{Z}[X]/(X^N + 1)$;
- $Q \gg q > 0$: the ciphertext moduli before (q) and after (Q) ModRaise;
- $\mathcal{R}_q$: the quotient ring $\mathcal{R}/q\mathcal{R}$, the plaintext space;
- Slots encoding encodes a message into a plaintext polynomial via the inverse complex canonical embedding.
- Coefficients encoding encodes a message in the coefficients of a plaintext polynomial.

The purpose of CKKS bootstrapping is to raise the FHE ciphertext modulus from q to $Q \gg q$ to enable further HE

computations, since the modulus is consumed after each homomorphic multiplication. It proceeds as follows.[3]

- SlotsToCoeffs: moves the messages from slots to coefficients, resulting in a ciphertext $(\mathbf{a}, \mathbf{b}) \in \mathcal{R}_q^2$ satisfying $\mathbf{b} = \mathbf{as} + \mathbf{m} + \mathbf{e} + q\mathbf{I}$ for some $\mathbf{I} \in \mathcal{R}$ and error $\mathbf{e}$, where $\mathbf{m}$ is the message,
- ModRaise: raises the ciphertext modulus from q to Q, resulting in $(\mathbf{a}', \mathbf{b}') \in \mathcal{R}_Q^2$, where $(\mathbf{a}', \mathbf{b}') = (\mathbf{a}, \mathbf{b}) \bmod q$, which satisfies $\mathbf{b} = \mathbf{as} + \mathbf{m} + \mathbf{e} + q\mathbf{I} \bmod Q$,
- CoeffsToSlots: moves the messages back to slots, having $\mathbf{m} + \mathbf{e}' + q\mathbf{I}$ in slots, where $\mathbf{e}'$ is a new error,
- EvalMod: homomorphically evaluates a polynomial, approximating the modulo q operation over an interval $[(-K+1)q, (K-1)q]$.

The last EvalMod step is to remove the unnecessary $q\mathbf{I}$ from the message, by (approximately) applying the modulo q operation. As it is a periodic function, we set a high-probability interval for the message $\mathbf{m} + \mathbf{e}' + q\mathbf{I}$, then approximate the function to a polynomial. The state-of-the art implementation uses Chebyshev approximations on the sine or cosine functions with a scaled interval, then applies double angle formula [11, 12, 29].

Inequality hints from CKKS bootstrapping failures. The multiplicative depth of EvalMod depends on the polynomial degree, thus, narrowing the approximation interval is preferred for bootstrapping performance. However, it raises the failing probability of EvalMod, which follows the Irwin-Hall distribution [36]. With probability $\Pr[|\mathbf{I}|_\infty > K]$, at least one slot contains a message that lies outside the approximation interval, which has a huge absolute value after the polynomial evaluation. If we decrypt the ciphertext after bootstrapping, the failed slot contains a message much larger than it should be. Based on the index i of the failed slot, we get information of $\mathbf{s}$ as

$$|(\mathbf{b} - \mathbf{as})_i| = |m_i + qI_i + e_i'| > K \ ,$$

where i is the index corresponding to the failed slot, which is a two-sided approximate inequality hint. The two-sided hints are already useful since the density function of the coefficient distributions with respect to $s_i = \pm 1$ are symmetric and convex, we can recover the indices of non-zero secret key coefficients.

However, since the polynomial approximating the modulo function is not symmetric with respect to the y-axis, by looking at the (sign of the) decrypted result, we can easily classify the failures and collect one-sided approximate inequality hints, each satisfying

$$\langle \mathbf{c}, \mathbf{z} \rangle + \frac{e_i'}{q} > K \ ,$$

[3]There is the other option for the orders of the subprocedures, starting from ModRaise and ending with SlotsToCoefficients, but this makes only little difference for our attack.

where $\mathbf{c} = (b_i, a_i, a_{i-1}, \cdots, a_0, -a_{N-1}, \cdots, -a_{i+1})/q$ and $\mathbf{z} = (1, \mathbf{s})$. We also note that $q \gg |m_i|_\infty \gg |e_i'|_\infty$, thus the error e_i'/q can mostly be ignored.

KR^{D} attack against CKKS. For the sake of simplicity, we focus on the RNS-CKKS FHE scheme and its Lattigo implementation [2] using sparse secret encapsulation [12]. In this scenario, the key switch operation is applied before and after ModRaise, to change the dense secret key (with a large Hamming weight) into a sparse secret key (with smaller Hamming weight), and vice versa. We note, our attack works for all of the CKKS variants and implementations that may or may not use the sparse secret encapsulation technique.

Our KR^{D} attack is given in Algorithm 6, which proceeds as follows. We first query an encryption of 0 and obtain a ciphertext ct, then query a bootstrapping on the it. We then query a decryption oracle to the ciphertext. From the decryption result, we find the slots that have messages whose absolute value is larger than a threshold δ, which is in general, sufficiently larger than usual errors. From the sign of the message, we can categorize the side of the failure. We collect the ciphertext ct, then we can evaluate SlotsToCoeffs and the sparse key encapsulation using the public key. By repeating this, we can collect a one-sided approximate inequality hint. The distribution of the coefficients in hints reveals the secret key if f is sufficiently large, thanks to Theorem 4.3.

Algorithm 6 KR^{D} attack on CKKS bootstrapping

$f := 0$
for $0 \le k < \gamma$ **do**
  Query $\mathsf{ct} \leftarrow \mathsf{Enc}(0)$
  Query $\mathsf{ct}' \leftarrow \mathsf{Bootstrap}(\mathsf{ct})$
  Query $m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}')$
  if $m_i > \delta$ for some $0 \le i \le N-1$ **then**
    Compute $\mathsf{ct}'' = (\mathbf{a}, \mathbf{b}) \leftarrow \mathsf{SparseEncap}(\mathsf{StC}(\mathsf{ct}))$
    Compute $\mathbf{c}_f := (a_i, a_{i-1}, \cdots, a_0, -a_{N-1}, \cdots, -a_{i+1})/q$
    $f := f + 1$
  end if
end for
Compute $\tilde{c}_i = \mathrm{Avg}(\{\mathbf{c}_{j,i} | \ \mathbf{c}_{j,i} > 0, j \in [f]\})$ for $i \in [N]$
Compute $s_{\mathrm{est},i} = \begin{cases} 1 & \text{if } \tilde{c}_i > \alpha/2, \\ -1 & \text{if } \tilde{c}_i < -\alpha/2, \\ 0 & \text{otherwise,} \end{cases}$ for $i \in [N]$
return $\mathbf{s}_{\mathrm{est}}$

In practice, the threshold δ can be easily chosen, since the decryption results are quite discretized and the polynomial approximating the modulo operation is public. For CKKS, the Hamming weight h of the secret key is public, so the categorization regarding the threshold α can be replaced by choosing the indices having the top h largest absolute values for $\tilde{c}_i$.

4.4 Experimental Results

Our KR^D attacks require at least a few ciphertexts that fails to decrypt correctly. Since even the fastest bootstrapping reports 10ms of running time in a moderate single-CPU, it requires more than a year to observe one failure for the default parameter sets.[4] These figures can be accelerated using GPUs or hardware accelerators [30, 45]. As the bootstrapping failure probability depends on the parameter sets, we first give the experimental results of the attacks targeting custom parameters with higher failure probabilities. We also provide experiments using the default parameters with simulated ciphertexts following the appropriate distribution. Then we give estimated results for other existing parameters.

DM/CGGI results. In OpenFHE [7], the decryption failure probabilities depicted in [41] range from 2^{-33} for the STD256 parameter set to 2^{-101} for the STD192Q parameter set.[5] In TFHE-rs [47], the decryption failure probability upper-bound is 2^{-40} in the default parameters. Since gate bootstrapping takes a relatively long time, of around 10ms on a single threaded CPU, performing 2^{40} gate bootstrapping may take more than hundreds of years. With the FPGA implementation from [8], this may still take around a year.

We highlight that this computation time is on the challenger side and that it is high only because of the limited performance of DM/CGGI. In fact, our KR^D attack is very efficient: it just averages the rounding errors corresponding to decryption failures.

Since current implementations of gate bootstrapping are too inefficient to experiment the attack on used parameter sets, we present two types of experimental results.

- *Attack for custom parameters*: We generate a custom parameter set with 128-bit IND-CPA security, but with higher failing probability. We collect the failing ciphertexts, then recover the secret key in several seconds using only the public APIs.
- *Attack using simulated ciphertexts for default parameters*: We simulate the ciphertexts conditioned on failures for the default parameters of TFHE-rs, using rejection sampling technique, then apply our key recovery attack. The simulation is based on the analysis given in Section 4.2. We reveal the secret key with less failing ciphertexts, as desired.

In Figure 4a, we give the accuracy of the secret key estimation using our KR^D attack, for the two parameters. The accuracy is given as the number of correctly estimated coefficients, say, $|\mathbf{s} - \mathbf{s}_{\text{est}}|_1$. Using our custom parameter set (see Appendix C for details), we run the KR^D experiment (Algorithm 5) with $\gamma = 1,000,000$ samples. The attack recovered 596 out of 600 secret key coefficients from $f = 8,434$ failing ciphertexts. We also run our attack on the TFHE-rs default parameters set DEFAULT_PARAMETERS, using the simulated ciphertext samples, and fully recover the secret key with less than 256 decryption failures. Note that he distribution of the rounding error coefficients of failed ciphertexts are already given in Figure 2a. We note that the two parameters enjoys the same level of IND-CPA security, estimated by the lattice estimator [4].[6]

CKKS results. In OpenFHE [7], the bootstrapping failure probability is set to lower than 2^{-22} in the default parameters, which can be induced from the Irwin-Hall distribution with approximation range parameter $K = 28$ and the Hamming weight $h = 128$ for the secret key. In Lattigo [2], the bootstrapping failure probability is set to lower than 2^{-138} in the default parameters, which is indeed 128-bit IND-CPA^D secure. However, prior works using Lattigo suggest the parameters with $\approx 2^{-16}$ [11] or $\approx 2^{-35}$ [12], which are not 128-bit IND-CPA^D secure. We note that each bootstrapping takes 30-50 seconds, thus collecting several failing slots will take at least a year. We highlight, again, that our KR^D attack takes only several tens of seconds to recover the secret key from the collected failing ciphertexts.

To run an experiment, we generate a custom parameter set with 128-bit IND-CPA security, but with higher failing probability. We collect the failing ciphertexts, then recover the secret key within several tens of seconds.

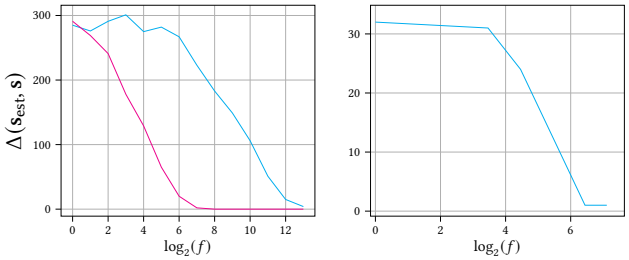
In Figure 4b, we give the accuracy of the secret key estimation for the custom parameters, which shows the number of correctly estimated +1's and −1's. We run the KR^D experiment (Algorithm 6) with $\gamma = 100$ samples. The attack recovered 31 out of 32 secret key coefficients. We note that, the attack outputs the 32 coefficients that has the highest averages among the N coefficients, however, we can also output several more coefficients in addition. In this case, with 87 failing slots, we can recover the full secret key from the top 42 coefficients, and with 138 failing slots, we can recover from the top 37 coefficients. Note that the distribution of the coefficients corresponds to the failed slots is already given in Figure 2b.

Other libraries and parameters. We summarize the parameter sets of DM/CGGI and CKKS implementations in the literature in Table 1. The third column shows the average slope for the distribution of the hints from the failing ciphertexts, for $s_i = 1$ (see Theorem 4.2 and Figure 2). The average of the slopes are given based on experiments for the custom parameters. For the rest of the parameters, the slopes are computed based on the parameters, or estimated based on the failing probabilities. The last column of the table shows the cost for collecting failing ciphertexts, in the number of

[4] e.g., for CGGI, DEFAULT_PARAMETERS of TFHE-rs takes $10\text{ms} \times 2^{40} \approx 350$ years, and for CKKS, a default parameter of OpenFHE takes $40\text{s} \times 2^{22} \approx 5$ years.

[5] One may generate custom parameters having lower failing probability with additional options, but we focus on their default settings.

[6] Git commit 564470e07d816f788d9c85acf72a1789c7787574



(a) CGGI **(b)** CKKS

Figure 4. Accuracy of the KR^{D} attack estimating the secret key from given number of failing ciphertexts (for CGGI) or slots (for CKKS). The cyan and magenta curves respectively correspond to the custom parameters (for both CGGI and CKKS) and the CGGI TFHE-rs DEFAULT_PARAMETERS. The x axis is the number of given failing ciphertexts (slots) in logarithm two, and the y axis is the accuracy of the secret key estimation, showing the number of incorrectly estimated coefficients.

Parameters	p_{fail}	Slope	Cost
DM/CGGI			
(Our) Custom	2^{-7}	1.21	2^{20}
OpenFHE [7] STD256	2^{-33}	1.39	$\approx 2^{46}$
TFHE-rs [47] DEFAULT	2^{-40}	2.48	$\approx 2^{50}$
OpenFHE [7] STD192Q	2^{-101}	2.89	$\approx 2^{111}$
TFHE-rs [47] TFHE_LIB	2^{-165}	5.75	$\approx 2^{171}$
CKKS			
(Our) Custom	0.87	$2^{4.1}$	2^{8}
[11]	2^{-16}	$2^{2.3}$	$\approx 2^{26}$
OpenFHE [7]$^{\#}$ [38]$^{\#}$, [20]*	2^{-22}	$2^{2.6}$	$\approx 2^{32}$
HEaaN [1], [12], [31]*, [25]*	2^{-35}	$2^{6.8}$	$\approx 2^{40}$
OpenFHE [7]$^{\#}$	2^{-57}	1.21	$\approx 2^{70}$
Lattigo [2], [12]	2^{-138}	$2^{9.0}$	$\approx 2^{143}$

Table 1. Parameter sets from the literature with claimed IND-CPA security larger than 128 bits. The columns show (upper bounds of) failure probabilities, slope for hint distributions, and estimated number of decryption queries for the attacks.

decryption queries. The numbers are estimated based on the failing probabilities and the average slopes. If the slope is enough steep, the secret key can be recovered with few, e.g., 32 ciphertexts. We note that recovering the secret key from the failing ciphertexts requires a few tens of seconds. For typical parameters of DM/CGGI, we denote their names.[7] For CKKS, most of the libraries use the same setting for their parameter sets, except for OpenFHE. It has two options for secret key distribution, sparse ternary with a fixed Hamming weight of 192 or uniform ternary. Asterisk (*) indicates the prior works using CKKS over discrete data [20, 25, 31]. All of them follows the libraries that use not sufficiently low failing probabilities. Pound (#) indicates the works claiming $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ security.

5 Conclusion

We exhibited $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ and KR^{D} attacks on homomorphic encryption schemes for exact data, when their correctness does not hold with probability sufficiently close to 1. These attacks also extend to threshold variants of those schemes. This work hence disproves the common belief that exact schemes would be immune to $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ attacks, oppositely to approximate schemes. Overall, what matters most for this notion of security is the correctness of the scheme rather than the type of data that it manipulates (although schemes on approximate data create specific definitional difficulties when it comes to correctness).

We emphasize that loose and possibly heuristic correctness suffices for IND-CPA security, which is relevant in all applications of homomorphic encryption where the output results are not shared. When the output results may be shared, correctness guarantees should be strengthened to thwart $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ attacks. This may lead to a performance penalty: for example, in the case of CKKS, a solution was given in [38] based on the noise flooding technique. To avoid paying for this performance penalty across all applications, a possibility is to specify in software whether the outputs may be shared or not [17].

[7] '_PARAMETERS' are omitted from the TFHE-rs parameters names.

References

[1] HEaaN private AI: Homomorphic encryption library. https://hub.docker.com/r/cryptolabinc/heaan. CryptoLab.Inc.

[2] Lattigo v5.0.2. https://github.com/tuneinsight/lattigo (commit 4cce9a48c1daaa2dd122921822f5ad70cd444156), Mar. 2024. EPFL-LDS, Tune Insight SA.

[3] E. Aharoni, N. Drucker, G. Ezov, E. Kushnir, H. Shaul, and O. Soceanu. E2E near-standard and practical authenticated transciphering. IACR Cryptol. ePrint Arch., 2023. https://eprint.iacr.org/2023/1040.

[4] M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015. Software available at https://github.com/malb/lattice-estimator.

[5] A. Alexandru, A. A. Badawi, D. Micciancio, and Y. Polyakov. Application-aware approximate homomorphic encryption: Configuring FHE for practical use. IACR Cryptol. ePrint Arch., 2024. https://eprint.iacr.org/2024/203.

[6] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, Apr. 2012.

[7] A. A. Badawi, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, Z. Liu, D. Micciancio, I. Quah, Y. Polyakov, R. V. Saraswathy, K. Rohloff, J. Saylor, D. Suponitsky, M. Triplett, V. Vaikuntanathan, and V. Zucca. OpenFHE: Open-source fully homomorphic encryption library. In *WAHC*, 2022. Library available at https://www.openfhe.org/, v1.1.1. (commit 4ebb28ea7bdd894a73bc5b73e59fcfbc7825330). IACR Cryptol. ePrint Arch. 2022/915, dated March 12, 2024.

[8] M. V. Beirendonck, J. D'Anvers, F. Turan, and I. Verbauwhede. FPT: A fixed-point accelerator for torus fully homomorphic encryption. In *CCS*, 2023.

[9] B. Biasioli, C. Marcolla, M. Calderini, and J. Mono. Improving and automating BFV parameters selection: An average-case approach. IACR Cryptol. ePrint Arch., 2023. https://eprint.iacr.org/2023/600.

[10] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Heidelberg, Aug. 2018.

[11] J.-P. Bossuat, C. Mouchet, J. R. Troncoso-Pastoriza, and J.-P. Hubaux. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In A. Canteaut and F.-X. Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 587–617. Springer, Heidelberg, Oct. 2021.

[12] J.-P. Bossuat, J. R. Troncoso-Pastoriza, and J.-P. Hubaux. Bootstrapping for approximate homomorphic encryption with negligible failure-probability by using sparse-secret encapsulation. In G. Ateniese and D. Venturi, editors, *ACNS 22*, volume 13269 of *LNCS*, pages 521–541. Springer, Heidelberg, June 2022.

[13] K. Boudgoust and P. Scholl. Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus. In J. Guo and R. Steinfeld, editors, *ASIACRYPT 2023, Part I*, volume 14438 of *LNCS*, pages 371–404. Springer, Heidelberg, Dec. 2023.

[14] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Heidelberg, Aug. 2012.

[15] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.

[16] M. Checri, R. Sirdey, A. Boudguiga, and J.-P. Bultel. On the practical CPA-D security of "exact" and threshold FHE schemes and libraries. IACR Cryptol. ePrint Arch., 2024. http://eprint.iacr.org/2024/116.

[17] J. H. Cheon, S. Hong, and D. Kim. Remark on the security of CKKS scheme in practice. IACR Cryptol. ePrint Arch., 2020. http://eprint.iacr.org/2010/1581.

[18] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song. Homomorphic encryption for arithmetic of approximate numbers. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 409–437. Springer, Heidelberg, Dec. 2017.

[19] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Heidelberg, Dec. 2016.

[20] H. Chung, H. Kim, Y.-S. Kim, and Y. Lee. Amortized large look-up table evaluation with multivariate polynomials for homomorphic encryption. Cryptology ePrint Archive, Paper 2024/274, 2024. https://eprint.iacr.org/2024/274.

[21] D. Dachman-Soled, L. Ducas, H. Gong, and M. Rossi. LWE with side information: Attacks and concrete security estimation. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 329–358. Springer, Heidelberg, Aug. 2020.

[22] D. Dachman-Soled, H. Gong, T. Hanson, and H. Kippen. Revisiting security estimation for LWE with hints from a geometric perspective. In H. Handschuh and A. Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 748–781. Springer, Heidelberg, Aug. 2023.

[23] M. Dahl, C. Danjou, D. Demmler, T. Frederiksen, P. Ivanov, M. Joye, D. Rotaru, N. Smart, and L. T. Thibault. fhEVM: Confidential EVM smart contracts using fully homomorphic encryption, 2023. https://github.com/zama-ai/fhevm/blob/main/fhevm-whitepaper.pdf.

[24] M. Dahl, D. Demmler, S. E. Kazdadi, A. Meyre, J. Orfila, D. Rotaru, N. P. Smart, S. Tap, and M. Walter. Noah's Ark: Efficient threshold-FHE using noise flooding. In *WAHC*, 2023.

[25] N. Drucker, G. Moshkowich, T. Pelleg, and H. Shaul. BLEACH: cleaning errors in discrete computations over CKKS. *J. Cryptol.*, 37(1):3, 2024.

[26] L. Ducas and D. Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640. Springer, Heidelberg, Apr. 2015.

[27] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption, 2012. http://eprint.iacr.org/2012/144.

[28] Q. Guo, D. Nabokov, E. Suvanto, and T. Johansson. Key recovery attack on approximate homomorphic encryption with non-worst-case noise flooding countermeasures. In *USENIX Security*, 2024.

[29] K. Han and D. Ki. Better bootstrapping for approximate homomorphic encryption. In S. Jarecki, editor, *CT-RSA 2020*, volume 12006 of *LNCS*, pages 364–390. Springer, Heidelberg, Feb. 2020.

[30] W. Jung, S. Kim, J. H. Ahn, J. H. Cheon, and Y. Lee. Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with GPUs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021.

[31] J. Kim, J. Seo, and Y. Song. Simpler and faster BFV bootstrapping for arbitrary plaintext modulus from CKKS. IACR Cryptol. ePrint Arch., 2024. https://eprint.iacr.org/2024/109.

[32] K. Kluczniak and G. Santato. On circuit private, multikey and threshold approximate homomorphic encryption. IACR Cryptol. ePrint Arch., 2023. https://eprint.iacr.org/2023/301.

[33] K. Kluczniak and G. Santato. On circuit private, multikey and threshold approximate homomorphic encryption. Cryptology ePrint Archive, Report 2023/301, 2023. https://eprint.iacr.org/2023/301.

[34] C. Knabenhans. Practical integrity protection for private computations, 2022. Available at https://pps-lab.com/student_theses/mthesis-christianknabenhans.pdf.

[35] A. Langlois and D. Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.*, 75(3):565–599, 2015.

[36] J.-W. Lee, E. Lee, Y. Lee, Y.-S. Kim, and J.-S. No. High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function. In A. Canteaut and F.-X. Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 618–647. Springer, Heidelberg, Oct. 2021.

[37] B. Li and D. Micciancio. On the security of homomorphic encryption on approximate numbers. In A. Canteaut and F.-X. Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 648–677. Springer, Heidelberg, Oct. 2021.

[38] B. Li, D. Micciancio, M. Schultz, and J. Sorrell. Securing approximate homomorphic encryption using differential privacy. In Y. Dodis and T. Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 560–589. Springer, Heidelberg, Aug. 2022.

[39] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May / June 2010.

[40] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. P. Fitzek, and N. Aaraj. Survey on fully homomorphic encryption, theory, and applications. *Proc. IEEE*, 2022.

[41] D. Micciancio and Y. Polyakov. Bootstrapping in FHEW-like cryptosystems. In *WAHC*, 2021. Available at https://eprint.iacr.org/2020/086. Version dated October 23, 2022.

[42] S. Murphy and R. Player. A central limit framework for ring-LWE decryption. Cryptology ePrint Archive, Report 2019/452, 2019. https://eprint.iacr.org/2019/452.

[43] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009.

[44] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public key encryption based on ideal lattices. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635. Springer,

Heidelberg, Dec. 2009.
[45] W. Wang, Y. Hu, L. Chen, X. Huang, and B. Sunar. Accelerating fully homomorphic encryption using GPU. In *HPEC*, 2012.
[46] Zama. Concrete: TFHE compiler that converts python programs into FHE equivalent, 2022. https://github.com/zama-ai/concrete (commit 9b2e39fd10b1e5172ab24a6fd561dc3603787643).
[47] Zama. TFHE-rs v0.4.1, 2023. https://docs.zama.ai/tfhe-rs (commit ad41fdf5a5060c0a981cd0c35bf998feafe68e02).

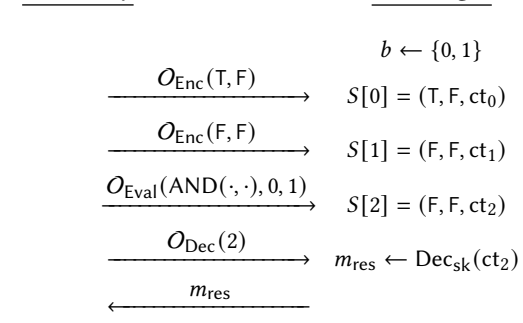
A A Generic IND-CPA$^{\mathsf{D}}$ Attack

In this section, we describe an IND-CPA$^{\mathsf{D}}$ attack against *non-perfectly correct* (F)HE schemes, i.e., FHE schemes with a non-zero decryption failure probability.

The attack simply consists in building ciphertexts which are supposed to result in m_0 or m_1 with $m_0 = m_1$, and then requesting their decryption. The decryption request is valid, as the underlying plaintext is supposed to be independent of b. The attack then exploits the fact that, even though the underlying plaintext messages are identical, the decryption failure probability of the corresponding ciphertexts can differ significantly. Since the adversary also knows the underlying plaintext, it can distinguish whether or not decryption failed, which leads to breaking IND-CPA$^{\mathsf{D}}$ security.

As a warm-up, we start with an attack targeting binary HE, i.e., HE whose plaintext space is $\{0, 1\}$. Then, we introduce a technique boosting the success probability with repetitions. We note that the attack can be easily extended to general HE schemes over larger rings, as $\{0, 1\}$ can be embedded in any ring.

Binary HE. Let $\{\mathsf{T}, \mathsf{F}\}$ denote the message space. We consider a binary HE scheme supporting Boolean operations AND and OR.



$$b' = \begin{cases} 0 & \text{if } m_{\mathsf{res}} = \mathsf{T} \ , \\ 1 & \text{else.} \end{cases}$$

Figure 5. A generic IND-CPA$^{\mathsf{D}}$ attack on binary HE.

Our IND-CPA$^{\mathsf{D}}$ attack, described in Figure 5, proceeds as follows. The attacker first makes two encryption queries ($m_0^0 = \mathsf{T}$, $m_1^0 = \mathsf{F}$) and ($m_0^1 = \mathsf{F}$, $m_1^1 = \mathsf{F}$). Then it asks the evaluation oracle to evaluate the AND function on the two ciphertexts, resulting in a ciphertext ct_2. Finally, it asks the decryption oracle to decrypt ct_2, and outputs $b' = 0$ if the decrypted result is T, and $b' = 1$ otherwise.

When the encryption and evaluation queries are given, the underlying messages are tracked, and stored in the shared state $S[0]$, $S[1]$, and $S[2]$, respectively. Since $\mathsf{AND}(m_b^0, m_b^1 = \mathsf{F})$ is always equal to F regardless of b, the tracked messages stored in $S[2]$ are F and F, and the decryption queries passes the check of the oracle. Hence, the attacker receives the decryption result $m_{\mathsf{res}} = \mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_2)$.

The decrypted result is $m_{\mathsf{res}} = \mathsf{F}$ if there is no failure during the homomorphic operations. However, if the ciphertext ct_1 fails to decrypt properly, i.e., $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_1) \neq \mathsf{F}$, then the decryption result of ct_2 is

$$\begin{aligned} \mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_2) &= \mathsf{AND}(\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_0), \mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_1)) \\ &= \mathsf{AND}(\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_0), \mathsf{T}) \\ &= m_b \ , \end{aligned}$$

assuming that no additional failure happened (which is the case with probability close to 1). The attacker thus has a success probability slightly higher than $1/2$ as:

$$\begin{aligned} \Pr[b = b'] &= (1-p) \cdot \Pr[b = b' \mid \neg \mathrm{A}] + p \cdot \Pr[b = b' \mid \mathrm{A}] \\ &= (1-p) \cdot \frac{1}{2} + p \cdot 1 \\ &= \frac{1}{2} + \frac{p}{2} \ , \end{aligned}$$

where A is the event $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_1) \neq \mathsf{F}$ and $p = \Pr[\mathrm{A}]$. We note this figure does not change when considering additional failures during the first encryption or the evaluation oracles.

Boosting the adversary's advantage. We can increase the success probability by repeating the encryption, evaluation, and decryption queries, and outputting 0 if and only if T appears among the decrypted results. This attack requires $(k+1)$ encryption queries, k evaluation queries and k decryption queries, and succeeds with probabiity

$$\begin{aligned} \Pr[b = b'] &\approx (1-p)^k \cdot \frac{1}{2} + \left(1 - (1-p)^k\right) \cdot 1 \\ &\approx \frac{1}{2} + \frac{kp}{2} \ , \end{aligned}$$

where the last approximation is valid when $kp \ll 1$.

We note that, by querying the circuit evaluating the OR of all the above AND operations, we can decrease the number of evaluation and decryption queries to a single one. Specifically, for a circuit $C : \mathcal{C}^{k+1} \mapsto \mathcal{C}$ defined as,

$$C(x, x_1, \cdots, x_k) = (x \wedge x_1) \vee (x \wedge x_2) \vee \cdots \vee (x \wedge x_k) \ ,$$

where $\wedge$ is an AND operation and $\vee$ is an OR operation, the two values $C(\mathsf{T}, \mathsf{F}, \cdots, \mathsf{F})$ and $C(\mathsf{F}, \mathsf{F}, \cdots, \mathsf{F})$ are both equal to F. However, if there exists i such that $x_i = \mathsf{T}$, the two values become different and will capture the failure. Thus, we can have the same advantage of approximately $1/2 + kp/2$ as before, but with only $(k+1)$ encryption, 1 evaluation, and

1 decryption queries. We note that this is a generalization ($k = 1$) of the attack in Figure 5, which queries k encryptions of $(m_0^i = \mathsf{F}, m_1^i = \mathsf{F})$ for $i = 1..k$ instead of one, and queries an evaluation of the circuit C instead of AND.

From fresh to arbitrary ciphertexts. We observe that the freshly encrypted ciphertexts ct_i (for $i = 1..k$) can be replaced by the ciphertexts having higher decryption failure probability. The circuit C can be replaced by a new composed one having larger depth, possibly with more encryption queries, but with the same number of evaluation and decryption queries.

For example, assume that there exists a circuit C^* with ℓ input wires such that $\mathsf{ct} = \mathsf{Eval}_{\mathsf{pk}}\left(C^*, (\mathsf{ct}^1, \cdots, \mathsf{ct}^\ell)\right)$ has a decryption failure probability p^*, over the randomness used by Enc to obtain the ct_j's (for $j = 1..\ell$). By composing the circuits C and C^* to $C' : C^{k\ell+1} \mapsto C$ defined as,

$$C' = C \circ (id, C^*, \cdots, C^*) \ ,$$

we increase the success probability to $\approx 1/2 + kp^*/2$, with $(N\ell + 1)$ encryption queries, 1 evaluation query, and 1 decryption query.

The discussion above shows that loose correctness directly impacts IND-CPA$^{\mathsf{D}}$ security. Further, in this context, what should be considered is the failure of decryption probability for a ciphertext resulting from the evaluation of a very large circuit.

B (In)security of Threshold-FHE

In this section, we discuss the insecurity of Threshold-FHE schemes by relying on our IND-CPA$^{\mathsf{D}}$ and KR$^{\mathsf{D}}$ attacks.

B.1 Definitions and Relation to IND-CPA$^{\mathsf{D}}$ Security

The purpose of Threshold-FHE is to distribute the secret decryption key among several parties, such that any subset of large enough size (corresponding to the threshold) can jointly decrypt any ciphertext, while any smaller subset of parties cannot learn anything about the underlying plaintext.

The capabilities of the attacker against Thres-IND-CPA security present important similarities with those in IND-CPA$^{\mathsf{D}}$ and KR$^{\mathsf{D}}$ security. Indeed, an attacker is allowed to request encryption queries, evaluation queries, as well as and partial decryption queries of the ciphertexts obtained from prior queries as long as the underlying plaintext is independent of the challenge bit b.

The main differences between Threshold-FHE and FHE are the setup procedure, which distributes the secret key between parties, and the decryption procedure, which is split in two phases: 1) each party can compute a partial decryption of a ciphertext using its share of the secret key, 2) (a sufficiently large set of) partial decryptions of a same ciphertext can be recombined to recover the underlying plaintext.

Since a threshold-FHE scheme encompasses a standard FHE scheme (e.g., consider the secret key as being the set of all partial decryption keys), our attacks extend to the case of threshold-FHE. However, we emphasize that in general, and unlike for standard FHE (for which many application scenarios do not require IND-CPA$^{\mathsf{D}}$ security), having access to a decryption oracle in scenarios involving threshold-FHE is the default option. Therefore, our attacks have a strong impact on threshold variants of the schemes studied so far.

Let us first recall the definition of Threshold-FHE.

Definition B.1 (Threshold-Fully Homomorphic Encryption)**.** Let $n \geq t \geq 0$. A (t, n)-threshold-fully homomorphic encryption scheme (Threshold-FHE) is a tuple of efficient algorithms (Setup, Enc, Eval, PDec, FinDec) with the following specifications:

- Setup outputs secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ and a public key pk;
- Enc takes as inputs a public key pk and a plaintext $m \in \{0, 1\}$, and outputs a ciphertext ct;
- Eval takes as inputs a public key pk, a binary circuit C, and a tuple of ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$ where k is the number of input wired of C, and outputs a ciphertext ct;
- PDec takes as inputs a secret key sk_i for $i \leq n$, and a ciphertext ct, and outputs a partial decryption p_i;
- FinDec takes as inputs a public key pk, and a set $\{\mathsf{p}_i\}_{i \in S}$ for some $S \subseteq \{1, \ldots, n\}$, and outputs a plaintext $m \in \{0, 1, \perp\}$.

For $\varepsilon \geq 0$ and a circuit size bound B, the scheme is said (ε, B)-correct if for any key set $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{pk})$ output by Setup, for any binary circuit C, for any plaintexts $m_1, \ldots, m_k \in \{0, 1\}$ where k is the number of input wires of C, for any set $S \subseteq \{1, \ldots, n\}$ with $|S| \geq t$, the following holds with probability $\geq 1 - \varepsilon$ over $\mathsf{ct}_j := \mathsf{Enc}_{\mathsf{pk}}(m_j)$ (for $j \leq k$) and $\mathsf{p}_i = \mathsf{PDec}_{\mathsf{sk}_i}\left(\mathsf{Eval}_{\mathsf{pk}}(C, (\mathsf{ct}_1, \ldots, \mathsf{ct}_k)\right)$:

$$\mathsf{FinDec}_{\mathsf{pk}}(\{\mathsf{p}_i\}_{i \in S}) = C(m_1, \ldots, m_k).$$

A scheme is perfectly correct if it is ε-correct for $\varepsilon = 0$.

As mentioned above, a threshold-FHE scheme Π encompasses an underlying FHE scheme Π^*, defined as:

- $\Pi^*.\mathsf{KeyGen}$ outputs $(\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n), \mathsf{pk})$, where $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{pk}) \leftarrow \Pi.\mathsf{Setup}$,
- $\Pi^*.\mathsf{Enc}_{\mathsf{pk}} = \Pi.\mathsf{Enc}_{\mathsf{pk}}$,
- $\Pi^*.\mathsf{Eval}_{\mathsf{pk}} = \Pi.\mathsf{Eval}_{\mathsf{pk}}$,
- $\Pi^*.\mathsf{Dec}_{\mathsf{sk}}(\cdot) = \Pi.\mathsf{FinDec}_{\mathsf{pk}}(\{\mathsf{PDec}_{\mathsf{sk}_i}(\cdot)\}_{i \leq n})$.

Thres-IND-CPA security is defined very similarly to IND-CPA$^{\mathsf{D}}$ security [32]. Actually, IND-CPA$^{\mathsf{D}}$ security can be precisely seen as Thres-IND-CPA security seeing the FHE scheme as a $(1, 1)$-Threshold FHE scheme. We provide a detailed definition below. Note that Thres-IND-CPA is an indistinguishability-based security definition. In the case of

threshold FHE, simulation security is often preferred, but similar indistinguishability-based security notions have been defined in various works on this topic [13, 33].

Definition B.2 (Thres-IND-CPA security)**.** Let $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PDec}, \mathsf{FinDec})$ denote a Treshold-FHE scheme. For an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ that is given access to the (stateful) oracles O_{Enc}, O_{Eval} and O_{PDec} defined below, we define the advantage $\mathsf{Adv}^{\text{Thres-IND-CPA}}(\mathcal{A})$ of $\mathcal{A}$ against the Thres-IND-CPA security of Π as:

$$\left| 2 \cdot \Pr\left[b = b' \;\middle|\; \begin{array}{c} (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{KeyGen}; \\ b \leftarrow \mathcal{U}(\{0,1\}); \\ \mathcal{T} \leftarrow \mathcal{A}_0(\mathsf{pk}); \\ b' \leftarrow \mathcal{A}_1^{O_{\mathsf{Enc}}, O_{\mathsf{Eval}}, O_{\mathsf{PDec}}}(\mathsf{pk}, \{\mathsf{sk}_i\}_{i \in \mathcal{T}}) \end{array} \right] - 1 \right|,$$

where $\mathcal{A}_0$ is required to output a (possibly empty) set $\mathcal{T} \subset \{1, \ldots, n\}$ of corrupted parties of size at most $t-1$. Oracles O_{Enc} and O_{Eval} are defined exactly as for IND-CPA$^{\mathsf{D}}$ security. Oracle O_{PDec} can be invoked only on a priorly obtained ciphertext. On input the index j of a priorly generated ciphertext, oracle O_{PDec} checks that the underlying plaintext is independent of the challenge bit b, and if so, returns $\mathsf{PDec}_{\mathsf{sk}_i}(\mathsf{ct}_j)$, for $i \leq n$.

Then, we have the following.

Theorem B.3. *Let Π be a Threshold-FHE scheme and Π^* be the underlying FHE scheme of Π. Let $\mathcal{B}$ an adversary against the* IND-CPA$^{\mathsf{D}}$ *security of Π^*. Then, there exists an adversary $\mathcal{A}$ against the* Thres-IND-CPA *security of Π, with same advantage and running time as $\mathcal{B}$.*

Proof. Let $\mathcal{B}$ be an adversary against the IND-CPA$^{\mathsf{D}}$ security of Π^*. We construct an adversary $\mathcal{A}$ against the security of Π. Adversary $\mathcal{A}$ obtains a public key pk from its challenger, which it forwards to $\mathcal{B}$. $\mathcal{A}$ does not corrupt any party (i.e., adversary $\mathcal{A}_0(\mathsf{pk})$ returns $\emptyset$). Now, adversary $\mathcal{A}_1$ runs $\mathcal{B}$ and makes the exact same queries as $\mathcal{B}$ to its own oracles. Then, for each query, adversary $\mathcal{A}_1$ simply forwards the response it obtains to $\mathcal{B}$, except for decryption queries. In that last case, it first reconstructs the actual decryption result by running FinDec on input the partial decryption shares it obtains from O_{PDec}, and then returns the result to $\mathcal{B}$. When $\mathcal{B}$ halts with some output bit b', so does $\mathcal{A}_1$.

By definition, all queries made by $\mathcal{B}$ are also valid queries for $\mathcal{A}_1$ since its queries must satisfy the exact same constraints (i.e., decryption queries should be made only for ciphertexts whose underlying plaintexts are independent of the challenge bit). Moreover, it can be seen that $\mathcal{A}$ correctly simulates an IND-CPA$^{\mathsf{D}}$ challenger in $\mathcal{B}$'s view, hence leading to our claim. □

B.2 Noah's Ark, a Threshold-FHE Scheme

We first recall the Threshold-FHE scheme from [24]. It builds upon the CGGI FHE scheme, and thus the underlying FHE scheme is very similar to the one we studied in Section 4.2.

The Setup stage can be done by using a secure multiparty protocol, generating the underlying secret key data in a secret-shared form. Once the keys are generated, the encryptions and evaluations can be done by anyone, with the public key. They are identical to the CGGI encryptions and evaluations. For a ciphertext $(b, \mathbf{a})$, the partial decryption algorithm computes $b - \langle \mathbf{a}, \mathbf{s}_i \rangle + e_i \bmod q$, where $\mathbf{s}_i$ is the secret key share of party P_i and e_i is a fresh error. The additional error is introduced to statistically hide the party's secret information: to obtain sufficient security, this error term is set quite large. The final decryption consists in computing a linear combination of shares.

As CGGI does not have the capacity to absorb a sufficiently large error e_i, in Noah's ark, the gap between the error and the plaintext message is increased by using a so-called Switch-and-Squash technique. It consists in switching the ciphertext modulus and the LWE dimension, squashing the errors via bootstrapping. The procedure is identical to CGGI bootstrapping, except that the moduli and dimension are changed: it takes as input an LWE ciphertext with dimension n and modulus q and outputs an LWE ciphertext with dimension n and modulus $2N$ via ModSwitch, then the ciphertext becomes an GLWE ciphertext of ring dimension N through BlindRotate, and finally goes back to an LWE ciphertext, now with dimension L and modulus Q.

B.3 Is There a Hole in Noah's Ark?

There are two possible sources of failures in Noah's Ark. A first one is the homomorphic evaluation of gates. A second one is the Switch-and-Squash method which involves a bootstrapping for a different set of parameters. In [24], parameter details are provided only for Switch-and-Squash, so we chose to present an attack targeting failures of PDec instead of Eval. The attack is described in Algorithm 7. For the sake of simplicity, we describe it for $n = t = 2$, even though this is not a parametrization considered in [24].

Algorithm 7 Attack on Noah's Ark, with $n = t = 2$.

$f := 0$
for $0 \leq k < \gamma$ **do**
  Query $\mathsf{ct}_k \leftarrow \mathsf{Enc}(\mathsf{true})$
  Query $\mathsf{ct}'_k \leftarrow \mathsf{Eval}(\mathsf{AND}, \mathsf{ct}_k, \mathsf{ct}_k)$
  Query $(\mathsf{p}_1, \mathsf{p}_2) \leftarrow \mathsf{PDec}_{\mathsf{sk}}(\mathsf{ct}_k)$
  Set $m \leftarrow \mathsf{FinDec}_{\mathsf{pk}}(\mathsf{p}_1, \mathsf{p}_2)$.
  if $m = \mathsf{false}$ **then**
    Compute the rounding error $\tilde{\mathbf{e}}_f$ from ct'_k
    $f := f + 1$
  end if
end for
Compute $\tilde{\mathbf{e}} = \frac{1}{f} \sum_{j=0}^{f-1} \tilde{\mathbf{e}}^j$
For all $i \leq n$, compute $\tilde{s}_i = \begin{cases} 1 & \text{if } \tilde{\mathbf{e}}_i < \alpha/2 \\ 0 & \text{otherwise} \end{cases}$
return $\mathbf{s}_{\text{est}}$

We recall in the columns of Table 2 the four parameter sets considered for Switch-and-Squash. The last row gives a lower bound on the decryption failure probability. The latter is possibly significantly higher as we compute the bound using only the ModSwitch error. Indeed, the other error terms cannot be obtained in [24].

ρ	1	4	1	4
(q, l)	$(2^{64}, 777)$	$(2^{64}, 870)$	$(2^{64}, 1024)$	$(2^{64}, 1024)$
(Q, L)	$(2^{128}, 4096)$	$(2^{128}, 4096)$	$(2^{128}, 4096)$	$(2^{128}, 4096)$
N'	1024	2048	1024	2048
w'	4	2	4	2
σ_{ms}	$2^{-8.49}$	$2^{-9.41}$	$2^{-8.29}$	$2^{-9.29}$
$2^{-\rho-1-1}$	2^{-3}	2^{-6}	2^{-3}	2^{-6}
r	$2^{5.49}$	$2^{3.41}$	$2^{5.29}$	$2^{3.29}$
$\text{erfc}(r/\sqrt{2})$	2^{-1460}	$2^{-85.1}$	2^{-1110}	$2^{-72.8}$

Table 2. Parameter sets of Switch-and-Squash and lower bounds of decryption failure probability. The notations are borrowed from [24].

C Custom Parameter Set for CGGI

Figure 6 shows the custom parameter that we use for CGGI attack, in TFHE-rs library.

Figure 6. Parameter sets of TFHE-rs with 128 bits of IND-CPA security.

Parameter	DEFAULT_PARAMETERS	CUSTOM_PARAMETERS
lwe_dimension	722	600
glwe_dimension	2	7
polynomial_size	512	128
lwe_modular_std_dev	0.000013072	0.000143792
glwe_modular_std_dev	0.000000050	0.000000550
pbs_base_log	6	6
pbs_level	3	3
ks_base_log	3	3
ks_level	4	4
encryption_key_choice	Small	Small