# Attacks Against the IND-CPA$^{D}$ Security of Exact FHE Schemes

Jung Hee Cheon*, Hyeongmin Choe†, Alain Passelègue‡, Damien Stehlé‡ and Elias Suvanto‡

*CryptoLab Inc., Seoul National University, Seoul, Republic of Korea
†Seoul National University, Seoul, Republic of Korea
‡CryptoLab Inc., Lyon, France

***Abstract*—A new 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 (erroneously) claiming that IND-CPA security and IND-CPA$^{D}$ security coincide for FHEs on exact data.**

**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 are not only exact but 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 indistinguishability and key-recovery attacks against all major exact FHE schemes. We illustrate their strength by concretely breaking the default BFV implementation of OpenFHE and simulating an attack for the default parameter set of the CGGI implementation of TFHE-rs (the attack is too expensive to be run on commodity desktops, because of the cost of CGGI bootstrapping). Our attacks extend to threshold versions of the exact FHE schemes, when the correctness is similarly loose.**

## 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 [1].

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. 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 [2] 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, with the restriction that the decryption oracle can be called only on ciphertexts whose underlying plaintexts are independent of the challenge bit.

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 it information that it already has. The definition from Li and Micciancio is motivated by the CKKS FHE scheme [3], 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) [4]. 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$^{D}$) attack. KR$^{D}$ security, later defined in [5], is an adaptation of IND-CPA$^{D}$ security that asks the attacker to recover the secret key. It is a weaker notion of security than IND-CPA$^{D}$ security, as a KR$^{D}$ attack gives an IND-CPA$^{D}$ attack.

Let us now consider exact FHE schemes, i.e., FHE schemes for discrete data with exact homomorphic eval-

uations. The most efficient such schemes are BGV [6], BFV [7, 8], DM [9] and CGGI [10], which are implemented in multiple libraries. Li and Micciancio claimed that FHE schemes for exact data are immune to IND-CPA$^{\mathsf{D}}$ attacks. This leads them to conclude that the IND-CPA$^{\mathsf{D}}$ security shows a weakness that would be specific to approximate FHE schemes. The same statement is repeated in [5].

**Our contributions.** We exhibit IND-CPA$^{\mathsf{D}}$ and KR$^{\mathsf{D}}$ attacks for BGV/BFV and DM/CGGI. This invalidates the claim of Li and Micciancio that exact schemes are immune to IND-CPA$^{\mathsf{D}}$ attacks and that IND-CPA$^{\mathsf{D}}$ insecurity would be specific to approximate schemes. As pointed out in [11] in the context of CKKS, IND-CPA$^{\mathsf{D}}$ security is a notion that rather models specific application scenarios of FHE. For many available implementations of the aforementioned exact schemes, our attacks are very efficient. We report experimental data showing their practical strength, breaking the BFV implementation from OpenFHE [12] and the CGGI implementation from TFHE-rs [13]. We stress that these are only examples, our attack being widely applicable. We also extend our attacks to some Threshold Fully Homomorphic Encryption schemes (Threshold-FHE). Threshold-FHE is an extension of FHE allowing to share the decryption key between different parties [14, 15]. As an illustration, we discuss the impact of our attack on the Noah's Ark scheme from [16].

*How is it even possible?* Li and Micciancio proved in [2, Lemma 1] that for an exact scheme, IND-CPA security implies IND-CPA$^{\mathsf{D}}$ security. The proof notably relies on the correctness of the FHE scheme, defined in [2, Section 2] by requiring that the decrypted value is correct with probability negligibly close to 1 as a function of the security parameter, over the internal randomness of the encryption algorithm. There are two significant caveats when translating this lemma into practice. First, for all efficient implementations of FHE schemes, correctness is heuristic. Statement 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 value of "negligible" may be sufficiently small for honest users but not for adversaries collecting decryption failures: concrete decryption failure "probabilities" can be of the order of $2^{-40}$ or even higher (for instance, Concrete-python, a TFHE compiler for TFHE-rs, has decryption failure probability of the order of $2^{-17}$). Our attacks exploit decryption failures. At a theoretical level, we propose a converse to [2, Lemma 1], whose proof implies that the advantages for IND-CPA and IND-CPA$^{\mathsf{D}}$ attackers differ by no more than the decryption error probability multiplied by the number of decryption queries. We give an IND-CPA$^{\mathsf{D}}$ attack whose advantage is essentially the decryption error probability multiplied by the number of decryption queries. Below, we focus on stronger attacks, namely KR$^{\mathsf{D}}$ attacks, for specific schemes and implementations.

*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 [17, 18] 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. The principle is to 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 [19] showed that those heuristic noise estimates can be exploited to obtain a KR$^{\mathsf{D}}$ attack against CKKS, by using correlated input ciphertexts. We show that such noise bounds also make BFV/BGV insecure by adapting the KR$^{\mathsf{D}}$ 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 the BFV implementation of OpenFHE.

*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 [4] formats for regular ciphertexts, and GLWE [6, 20] 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 [13]. For its default parameters, the bootstrapping failure probability is of the order of $2^{-40}$. We then expect to have to run around $2^{40}$ bootstraps to observe a bootstrapping error. This is large (but not unreachable) when taking into account the limited performance of TFHE-rs bootstrapping: it would take around 300 years on a single-thread CPU. We consider two experiments. 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 error probability of around $1\%$. In this case, we are able to mount our attack 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 256 bootstrapping failures.

*Extension to Threshold-FHE.* Threshold-FHE, introduced in [14] and then formalized and generalized in [15], is an extension of FHE where the decryption key is shared between different users. To decrypt, the users locally run a partial decryption and then recombine their partial decryption results to obtain a plaintext. An adversary is allowed to corrupt some number of parties and to access a partial decryption oracle for the other parties. In the indistinguishability-based security definition, it is then required to distinguish between two sets of ciphertexts, with the restriction that the partial decryption oracle calls for corresponding ciphertexts in the two sets must correspond to identical plaintexts. To a threshold-FHE scheme, we associate the FHE scheme with the same key generation, encryption and evaluation algorithms, and with decryption defined as the composition of the partial decryptions and recombination. Then one can see that if there is an IND-CPA$^{\mathsf{D}}$ attack on the FHE scheme, it is possible to derive an attack on the Threshold-FHE scheme. Given this, it is tempting to extend our IND-CPA$^{\mathsf{D}}$ attacks to concrete proposals of Threshold-FHE schemes. We detail a concrete attack on the Noah's scheme from [16], extending the above IND-CPA$^{\mathsf{D}}$ attack on TFHE-rs. We stress that we could as well have chosen other Threshold-FHE schemes derived from BGV/BFV and DM/CGGI.

The codes are publicly available at the following git repository:

https://github.com/hmchoe0528/INDCPAD_HE_ThresFHE

## 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 $U(X)$ 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 $\mu$ with standard deviation parameter $\sigma$ 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 $\mu$ and standard deviation parameter $\sigma$, 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 $\mu$ when it is 0.

For the normal distribution over $\mathbb{R}$, centered at $\mu$ and with standard deviation $\sigma$, 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 $t > 0$:

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

### 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* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *with the following specifications:*

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

*For* $\varepsilon \geq 0$ *and a circuit size bound* $B$*, the scheme is said* $(\varepsilon, B)$*-correct if for any pair* $(\mathsf{sk}, \mathsf{pk})$ *output by* $\mathsf{KeyGen}$*, 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* $C$*, the following holds with probability* $\geq 1 - \varepsilon$ *over* $\mathsf{ct}_i := \mathsf{Enc}(\mathsf{pk}, m_i)$ *(for* $i \leq k$*):*

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

*An FHE scheme is perfectly correct if it is* $\varepsilon$*-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 $\varepsilon$-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 $\varepsilon$-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)**.** *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *be an FHE scheme. We define the advantage* $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}(\mathcal{A})$ *of an adversary* $\mathcal{A}$ *against the* IND-CPA *security of* $\Pi$ *as:*

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

In the context of FHE, IND-CPA$^{\mathsf{D}}$ security is typically inherited from the presumed hardness of LWE [4], RLWE [17, 18], GLWE [6, 20] and circular security assumptions.

## 2.2. IND-CPA$^{\mathsf{D}}$ Security

IND-CPA$^{\mathsf{D}}$ security, introduced in [2], 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 $\mathcal{O}_{\mathsf{Enc}}(m_0, m_1; \mathsf{pk}, b, i)$

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

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

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

**Algorithm 3** Decryption oracle $\mathcal{O}_{\mathsf{Dec}}(j; \mathsf{sk}, S)$

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

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

## 2.3. KR$^{\mathsf{D}}$ Security

KR$^{\mathsf{D}}$ security is a relaxation of IND-CPA$^{\mathsf{D}}$ security introduced in [5]. 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$^{\mathsf{D}}$ security)**.** *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *be an FHE scheme. For an adversary* $\mathcal{A}$ *that is given access to the (stateful) oracles* $\mathcal{O}_{\mathsf{Enc}}$*,* $\mathcal{O}_{\mathsf{Eval}}$ *and* $\mathcal{O}_{\mathsf{Dec}}$ *with* $b$ *fixed to* $0$*, we define the success probability* $\mathsf{Succ}^{\mathsf{KRD}}(\mathcal{A})$ *against the* KR$^{\mathsf{D}}$ *security of* $\Pi$ *as:*

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

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

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

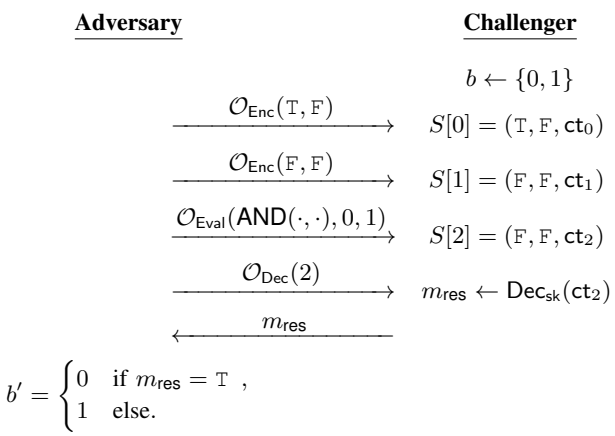
# 3. A Generic IND-CPA$^{\mathsf{D}}$ Attack

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 $\mathsf{IND\text{-}CPA^D}$ 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 $\{\mathtt{T},\mathtt{F}\}$ denote the message space. We consider a binary HE scheme supporting Boolean operations $\mathsf{AND}$ and $\mathsf{OR}$.




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

Figure 1: A generic $\mathsf{IND\text{-}CPA^D}$ attack on binary HE.

Our $\mathsf{IND\text{-}CPA^D}$ attack, described in Figure 1, proceeds as follows. The attacker first makes two encryption queries $(m_0^0 = \mathtt{T},\ m_1^0 = \mathtt{F})$ and $(m_0^1 = \mathtt{F},\ m_1^1 = \mathtt{F})$. Then it asks the evaluation oracle to evaluate the $\mathsf{AND}$ function on the two ciphertexts, resulting in a ciphertext $\mathsf{ct}_2$. Finally, it asks the decryption oracle to decrypt $\mathsf{ct}_2$.

Since $\mathsf{AND}(m_b^0, \mathtt{F})$ is always equal to $\mathtt{F}$ regardless of $b$, the tracked message are $\mathtt{F}$ and $\mathtt{F}$, 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 attacker outputs $b' = 0$ if the decrypted result if $\mathtt{T}$, and $b' = 1$ otherwise.

The decrypted result is $m_{\mathsf{res}} = \mathtt{F}$ if there is no failure during the homomorphic operations. However, if the ciphertext $\mathsf{ct}_1$ fails to decrypt properly, i.e., $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_1) \neq \mathtt{F}$, then the decryption result of $\mathsf{ct}_2$ is

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

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 $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_1) \neq \mathtt{F}$ and $p = \Pr[\mathrm{A}]$.

**Boosting the adversary's advantage.** We can increase the success probability by repeating the encryption, evaluation, and decryption queries

$$\begin{gathered} \mathcal{O}_{\mathsf{Enc}}(m_0^i = \mathtt{F}, m_1^i = \mathtt{F}), \\ \mathcal{O}_{\mathsf{Eval}}(\mathsf{AND}(\cdot,\cdot), 0, 2i-1), \\ \mathcal{O}_{\mathsf{Dec}}(2i), \end{gathered}$$

for $i = 1, \ldots, N$, and the attacker receives $N$ decrypted messages

$$\forall i \leq N : \ m_{\mathsf{res},i} = \mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_{2i}).$$

The adversary outputs 0 if $\mathtt{T}$ appears among the decrypted results, and else outputs 1. This attack requires $(N+1)$ encryption queries, $N$ evaluation queries and $N$ decryption queries, and succeeds with probabiity

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

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

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

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

where $\wedge$ is an $\mathsf{AND}$ operation and $\vee$ is an OR operation, the two values $C(\mathtt{T},\mathtt{F},\cdots,\mathtt{F})$ and $C(\mathtt{F},\mathtt{F},\cdots,\mathtt{F})$ are both equal to $\mathtt{F}$. However, if there exists $i$ such that $x_i = \mathtt{T}$, the two values become different. Thus, we can have the same advantage of approximately $1/2 + Np/2$ as before, but with only $(N+1)$ encryption, 1 evaluation, and 1 decryption queries. We detail this attack in Figure 2, which can be viewed as a generalization $(N = 1)$ of the attack in Figure 1.

**From fresh to arbitrary ciphertexts.** We observe that the freshly encrypted ciphertexts $\mathsf{ct}_i$ (for $i \leq N$) can be replaced by the ciphertexts having higher decryption failure probability.

For example, assume that there exists a circuit $C^*$ with $k$ input wires such that $\mathsf{ct} = \mathsf{Eval}_{\mathsf{pk}}(C^*, (\mathsf{ct}^1, \cdots, \mathsf{ct}^k))$ has a decryption failure probability $p^*$, over the randomness used by $\mathsf{Enc}$ to obtain the $\mathsf{ct}_j$'s (for $j \leq k$) . Then we replace the

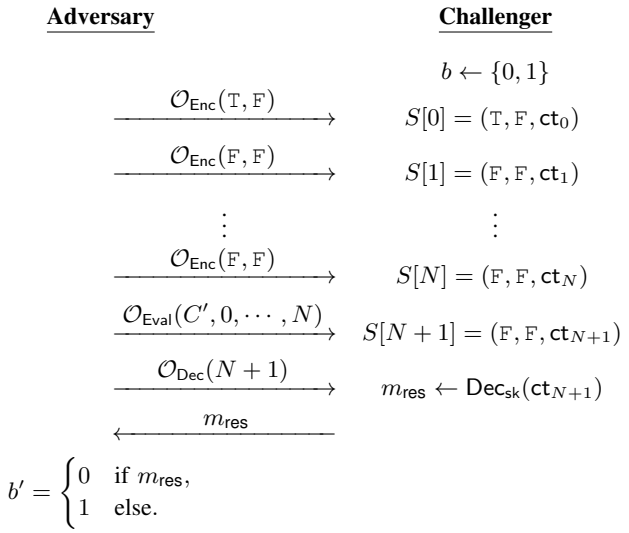



Figure 2: Boosted IND-CPA$^{\mathsf{D}}$ attack on binary HE.

$(N+1)$ encryption queries by $(Nk+1)$ encryption queries and change the evaluation circuit $C : \mathcal{C}^{N+1} \mapsto \mathcal{C}$ by

$$C' = C \circ (id, C^*, \cdots, C^*) : \mathcal{C}^{Nk+1} \mapsto \mathcal{C} \ .$$

This attack can be viewed as a generalization of the attack in Figure 2 (by taking $C^* = \mathsf{id}$ and $k = 1$). It can be mounted with $(Nk+1)$ encryption queries, 1 evaluation query, and 1 decryption query. The success probability of this attack is $\approx 1/2 + Np^*/2$, i.e., exactly the same as before, but $p$ is replaced by $p^*$.

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.

## 4. KR$^{\mathsf{D}}$ Attacks on B(F/G)V-like FHE Schemes

In this section, we present our KR$^{\mathsf{D}}$ attack against the BGV and BFV (F)HE schemes [6, 7, 8] 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. 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 [12, 23, 24] rely on average-case noise analyses to improve performance. These are vulnerable to our attack.

### 4.1. 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;
- $\Delta = \lfloor q/t \rfloor$: 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 $\mathsf{ct} = (\mathbf{b}, \mathbf{a}) \in \mathcal{R}_q^2$ 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 $D_\sigma$, a discrete Gaussian distribution over $\mathbb{Z}$ centered in 0 with standard deviation $\sigma$. For two ciphertexts, we define their homomorphic addition as their component-wise addition in $\mathcal{R}_q^2$, i.e., for $\mathsf{ct}_0 = (\mathbf{b}_0, \mathbf{a}_0)$ and $\mathsf{ct}_1 = (\mathbf{b}_1, \mathbf{a}_1)$ 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 $D_\sigma$. However, if the two ciphertexts are correlated, the variance can be smaller, or larger. In the worst case, we have $\mathsf{ct}_0 = \mathsf{ct}_1$ 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 $\sigma$ is the standard deviation of the error in the ciphertext under scope.

### 4.2. 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.

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 3. For the sake of simplicity, we focus on the power-of-2 integer for scale factor $\Delta$. We first encrypt 0 and obtain a ciphertext $\mathsf{ct}_0$, whose error $\mathbf{e}$ is sampled from $D_\sigma$. We then recursively add the ciphertext to itself. The standard deviation of the resulting error is doubled for each iteration. After $k = \lfloor \log_2 \Delta \rceil$ iterations, the error

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

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

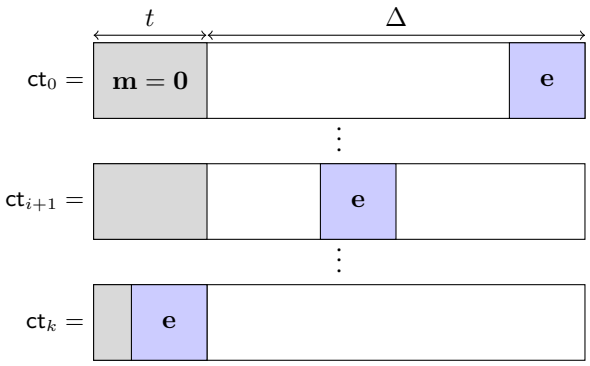



Figure 3: Position of the error in $\mathbf{b} + \mathbf{as} \bmod q$ during the iterations of the $\mathsf{KR}^{\mathsf{D}}$ attack.

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 estimation of the decryption failure probability based on the average-case analysis is $\approx \mathrm{erfc}((\Delta/2)/(\sqrt{2\Delta}\sigma))$. Unless this is deemed too large, the attacker receives the decryption result by querying the decryption oracle, and then obtains

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

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 $\lfloor \log_2 \Delta \rceil - 1$, $\lfloor \log_2 \Delta \rceil - 2$, etc iterations instead of $\lfloor \log_2 \Delta \rceil$, 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.

### 4.3. Experimental Results

For our experiments, we consider a typical BFV parameter set for the OpenFHE library. We have $N = 4096$, $q = 2^{60}$, $t = 2^{16}+1$, and $\Delta = 2^{44} - \varepsilon$ for some small $\varepsilon$. 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. 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 $\approx 2^7$.

## 5. A $\mathsf{KR}^{\mathsf{D}}$ Attack on DM/CGGI

In this section, we present our $\mathsf{KR}^{\mathsf{D}}$ attack against the DM and CGGI FHE schemes [9, 10], also respectively known as FHEW and TFHE. For the sake of simplicity, we only focus on the CGGI scheme and the TFHE-rs library [13], but stress that our attack is general and not specific to our choice of variant for DM/CGGI and our choice of target implementation. As in the $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ attacks of Section 3, the $\mathsf{KR}^{\mathsf{D}}$ attacker can observe decryption failures and obtain information from the failure/non-failure events. Our $\mathsf{KR}^{\mathsf{D}}$ attack takes advantage of the large rounding error in the ModSwitch step during gate bootstrapping, that gives approximate hints on the secret key.

In the following, we first recall the CGGI (gate) bootstrapping. We then introduce the $\mathsf{KR}^{\mathsf{D}}$ attack strategy and, finally, we illustrate its strength with experiments.

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

**TFHE bootstrapping.** Let us start with an LWE ciphertext $\mathsf{ct} = (b, a_1, \ldots, a_n) \in \mathbb{Z}_q^{n+1}$ 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 $e_{\mathsf{in}}$ whose absolute value is small relative to $\Delta$. The bootstrapping procedure consists of 4 steps, namely, ModSwitch, BlindRotate, SampleExtract, and KeySwitch. An illustration of bootstrapping is provided in Figure 4.

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.

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

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

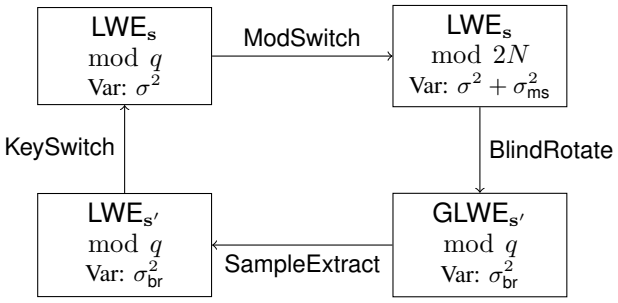



Figure 4: High-level overview of the TFHE bootstrapping loop. In each box, we give the ciphertext format, the secret key, the current ciphertext modulus, and the (heuristic) normalized variance of the noise after decryption with the secret key (the normalized variance is the variance divided by the square of the modulus). The initial variance $\sigma^2$ can vary depending on the input: a fresh ciphertext, a bootstrapped ciphertext, or a linear combination of bootstrapped ciphertexts (for gate bootstrapping).

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

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

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

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

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.

The BlindRotate step computes $X^{-\tilde{b}+\sum_i \tilde{a}_i s_i} \cdot V$ in encrypted state, using the GLWE encryptions of the $s_i$'s under the GLWE secret key $\mathbf{s}'$. The polynomial $V$ allows to map the message in the exponent to the constant term. Additional errors are introduced during this step in the coefficients of the GLWE ciphertexts, whereas the exponent of the plaintext monomial is handled exactly. Thus, the pre-BlindRotate errors and the post-BlindRotate are (heuristically) independent.

The last two steps of the bootstrapping loop are SampleExtract, which extracts the constant term of the GLWE ciphertext to make an LWE ciphertext under the secret key set $\mathbf{s}'$ (converted from polynomial to vector), and KeySwitch, which changes the secret key $\mathbf{s}'$ to the original one $\mathbf{s}$. The SampleExtract step does not introduce any additional errors, but the KeySwitch step does.

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

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

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. For example, for $\mathsf{ct}_0$ and $\mathsf{ct}_1$ two ciphertexts modulo $q$ respectively decrypting to $m_0$ and $m_1$ under key $\mathbf{s}'$, the AND gate is implemented by computing $\mathsf{ct} = \mathsf{ct}_0 + \mathsf{ct}_1 - (q/8, 0, \cdots, 0)$. The

ciphertext ct then goes through ModSwitch, BlindRotate, etc. If $\mathsf{ct}_0 = \mathsf{ct}_1$, we have $\sigma^2 = 4(\sigma_{\mathsf{br}}^2 + \sigma_{\mathsf{ks}}^2)$ and thus $\sigma_{\mathsf{gbts}}^2 = 4\sigma_{\mathsf{br}}^2 + 4\sigma_{\mathsf{ks}}^2 + \sigma_{\mathsf{ms}}^2$. Assuming that the error behaves as a continuous Gaussian, the probability of incorrect bootstrapping is $\mathrm{erfc}(1/(16\sqrt{2}\cdot\sigma_{\mathsf{gbts}}))$.

As an example, consider the DEFAULT_PARAMETERS of the TFHE-rs library. We have $n = 722$, $N = 512$ and $q = 2^{32}$. The claimed error probability is at most $2^{-40}$.[1] The error in the pre-ModSwitch ciphertexts has an experimentally measured standard deviation of $\sqrt{\sigma_{\mathsf{br}}^2 + \sigma_{\mathsf{ks}}^2} = 2^{-8.31}$. The ModSwitch error has a relative standard deviation of $\sigma_{\mathsf{ms}} = 2^{-7.54}$. We then obtain a somewhat lower than claimed probability of incorrect gate bootstrapping of $2^{-44.41}$.

## 5.2. Key-Recovery Attack

The attacker has access to a decryption oracle (limited to properly created ciphertexts). Decryption takes place for ciphertexts modulo $q$ under key $\tilde{\mathbf{s}}$, i.e., after KeySwitch and before ModSwitch. The attack consists in generating a ciphertext, gate-bootstrapping it a first time to increase its error, then gate-bootstrapping it again and checking if it decrypts properly. Decryption can fail for several reasons:

- the post-ModSwitch error in the first bootstrap is too large;
- the post-ModSwitch error in the gate-bootstrap is too large;
- the pre-decryption BlindRotate error is too large.

As the bootstrapping error is much smaller for a fresh ciphertext than for a bootstrapped ciphertext, the first possibility is much less likely than the second one. Similarly, the error due to BlindRotate is small compared to the overall bootstrapping error, making the third event very unlikely compared to the second one. Finally, the conjunction of more than one such event is very unlikely. In particular, we can neglect the event that two such events could compensate themselves and result in a correct ciphertext. Overall, when decryption fails in our process, it indicates, with probability extremely close to 1, that the post-ModSwitch error in the gate-bootstrap was above the correct decryption threshold. Choosing the plaintext and gate appropriately, the latter is $1/16$ (after normalization by the modulus).

In the case of decryption failure, ModSwitch also provides an error vector $\tilde{\mathbf{e}} \in \mathbb{Z}^{n+1}$, giving information of the form

$$\langle \tilde{\mathbf{e}}, \tilde{\mathbf{s}} \rangle + e > t \ ,$$

for some unknown secret $\tilde{\mathbf{s}} = (1, \mathbf{s})$, error $e = 2Ne_{\mathsf{in}}/q$ and a known threshold $t = 2N/16$. This leads us to define approximate inequality hints on a secret key.

**Definition 5.1** (Approximate Inequality Hint)**.** *For $\sigma, t > 0$, a $(\sigma, t)$-approximate-inequality hint on the secret key $\mathbf{s} \in \{0,1\}^n$ is a vector $\mathbf{e} \in \mathbb{R}^n$ such that*

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

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

In the case of ModSwitch, the error vector $\tilde{\mathbf{e}}$ is uniformly distributed in $(-1/2, 1/2]^{n+1}$, but a decryption failure tells us whether a given $\tilde{\mathbf{e}}$ is an approximate inequality hint. As a result, those $\tilde{\mathbf{e}}$'s are well-aligned with the secret key $\tilde{\mathbf{s}}$. This leads us to the following statement, on the distribution of rounding error coefficients conditioned on failures.

**Lemma 5.2.** *Let $\sigma, t > 0$ and a secret key $\tilde{\mathbf{s}} \in \{0,1\}^n$. Let $\tilde{\mathbf{e}} \sim \mathcal{U}([-1/2, 1/2]^n)$ and $e \sim \mathcal{N}(0, \sigma^2)$. Let $i \le n$ and $Y_i = \langle \tilde{\mathbf{e}}, \tilde{\mathbf{s}} \rangle - \tilde{e}_i s_i$. The probability density function $f$ of $\tilde{\mathbf{e}}_i$ conditioned on the event $\langle \tilde{\mathbf{e}}, \tilde{\mathbf{s}} \rangle + e > t$ satisfies, for all $x_i \in [-1/2, 1/2]$:*

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

*Proof.* We consider two cases. If $s_i = 0$, the failure event is independent of the random variable $\tilde{e}_i$ and its conditional distribution remains the uniform distribution on $[-1/2, 1/2]$. If $s_i = 1$, Bayes' law gives

$$f(x_i) = 1 \cdot \frac{\Pr[\langle \tilde{\mathbf{e}}, \tilde{\mathbf{s}} \rangle + e > t | \tilde{\mathbf{e}}_i = x_i]}{\Pr[\langle \tilde{\mathbf{e}}, \tilde{\mathbf{s}} \rangle + e > t]} \ .$$

We use the equality $\tilde{e}_i s_i + Y_i = \langle \tilde{\mathbf{e}}, \tilde{\mathbf{s}} \rangle$ to conclude. □

In our case, as $n$ is very large, we can heuristically assume that $Y_i$ has a distribution similar to that of $\langle \tilde{\mathbf{e}}, \tilde{\mathbf{s}} \rangle$. Lemma 5.2 implies that the distributions of the rounding error coefficients $\tilde{e}_i$ are very different depending on their corresponding secret key coefficient $s_i$. Indeed, one is uniform while the other is similar to a truncated scaled erfc function. As $t$ is chosen so that the gate bootstrapping failure probability is low, we are indeed somewhat far in the erfc function, in a range where it varies significantly. Figure 5 displays the distribution of $\tilde{e}_i$ for the TFHE-rs DEFAULT_PARAMETERS. We note that we could similarly exploit the distributions of the $\tilde{e}_i$'s conditioned on correct decryption, but then the two distribution functions, for $s_i = 0$ and $s_i = 1$, are extremely close.

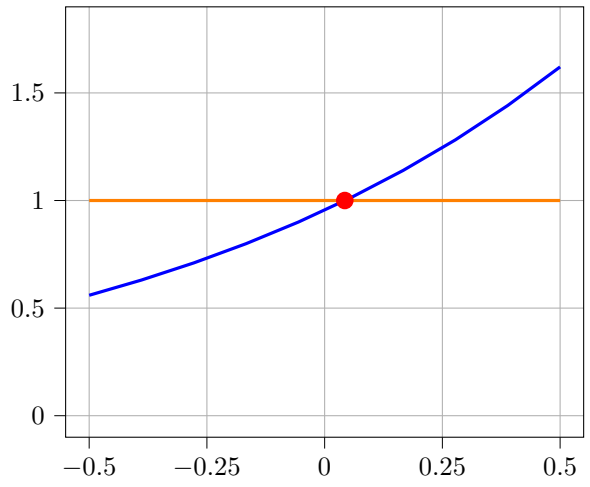
The attack exploits the fact that the two pdfs differ significantly. In particular, the expectancy of the one corresponding to $s_i = 0$ is 0, whereas the other one has expectancy $\alpha > 0$. Intuitively, if a value $\tilde{e}_i$ is below $\alpha/2$, it is likely that it was picked from the pdf corresponding to $s_i = 0$, whereas if it is above $\alpha/2$, it is likely that it was picked from the pdf corresponding to $s_i = 1$. This leads to the attack described in Algorithm 6. The calls to Enc, Eval and Dec formally correspond to oracles to the challenger, as per Definition 2.4. The parameter $\gamma$ quantifies the number of attempts to create decryption failures. The variable $f$ counts the number of observed decryption failures. Finally, the vector $\mathbf{s}_{\mathsf{est}}$ is a guess for the secret key $\mathbf{s}$.

*Claim.* Algorithm 6 recovers the secret key $\mathbf{s}$ with overwhelming probability, if the number of trials $f$ satisfies $f = \Omega(\sqrt{\log n}/\alpha^2)$.

Indeed, the random variable $\tilde{\mathbf{e}} = (\sum_{j=0}^{f-1} \tilde{\mathbf{e}}^j)/f \in \mathbb{R}^n$ is a sum of i.i.d. random variables. The central limit theorem

1. This figure is borrowed from the TFHE-rs library and its website, https://docs.zama.ai/tfhe-rs/fine-grained-apis/boolean/parameters.

Figure 5: Distributions of the coefficients of $\tilde{\mathbf{e}}$ conditioned on decryption failures, for the TFHE-rs library DEFAULT_PARAMETERS. The blue and orange curves respectively depict the pdfs of $\tilde{e}_i$ when $s_i = 1$ and $s_i = 0$.




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

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

states that its distribution is close to an $n$-dimensional Gaussian distribution, when $f$ is sufficiently large. For the coordinates $i$ such that $s_i = 0$, we have a sum of i.i.d. uniformly random variables over $[-0.5, 0.5]$, so that the standard deviation is $\sigma_0 = 1/\sqrt{12f}$ and the center is 0. For the coordinates $i$ such that $s_i = 1$, the center is $\alpha$ and the standard deviation is some $\sigma_1 = \Theta(1/\sqrt{f})$. Therefore, for each $i \le n$, the reply of Algorithm 6 is incorrect with probability $\le \mathrm{erfc}(O(\alpha\sqrt{f})) = O(1/n)$.

### 5.3. Experimental Results

The $\mathsf{KR}^{\mathsf{D}}$ attack requires sufficiently many gate bootstraps to observe decryption failures. The gate bootstrapping failure probability depends on the considered parameter set. In OpenFHE [12], the decryption failure probabilities mentioned in depicted in [25] range from $2^{-33}$ for the STD256 parameter set to $2^{-101}$ for the STD192Q parameter set. In TFHE-rs [13], the decryption failure probability upper-bound is $2^{-40}$ in the default parameters. The Hamming weight of the secret key increases the ModSwitch error, so the failure probability depends on the secret key. 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 300 years. With the FPGA implementation from [26], this may still take around a year. We highlight that this computation time is on the challenger/oracle side and that it is high only because of the limited performance of DM/CGGI. In fact, our $\mathsf{KR}^{\mathsf{D}}$ 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.

- TFHE-rs *with modified parameters*. We increased the error in the switching keys used in BlindRotate and decreased the ring dimension $N$. This allows to maintain the same level of IND-CPA security, and at the same time increase the bootstrapping error probability. It also provides better bootstrapping performance. The bootstrapping error probability and the efficiency of bootstrapping allow to experiment the attack in practice in a reasonable time on a conventional CPU. Our experiments show that our attack recovers the secret key in practice, using only the public API. The challenger side is executed by a Rust code using TFHE-rs while the $\mathsf{KR}^{\mathsf{D}}$ adversary is implemented in Python.
- *Simulations*. We simulate the distribution of ciphertexts conditioned on failures using a standard rejection sampling technique. This allows us to circumvent the slowness of current implementations and to experiment our attack on more parameter sets, in particular those with lower decryption failure probabilities.

**Practical experiments.** Table 6 gives the default parameters of TFHE-rs for 128 bits of IND-CPA security and our modified parameters, which still enjoy 128 bits of IND-CPA security. As usual in FHE literature, the bit security is computed by the lattice estimator [27].[2]

Note that security increases with the dimension and the relative error. We decreased the dimension and increased the relative noise to maintain security. This results both in a performance increase and an increased failure probability. The custom parameter set has a decryption failure probability of the order of 1%.

Using our custom parameter set, we ran the $\mathsf{KR}^{\mathsf{D}}$ experiment (Algorithm 6) with $\gamma = 1,000,000$ samples. We plotted the experimental distribution of the rounding error coefficients of failed ciphertexts. The blue (resp. orange) histogram depicts the experimental distribution of the rounding error coefficients $\tilde{e}_i$ such that $s_i = 1$ (resp. $s_i = 0$). The figure shows that even with only a single failure, the advantage for recovering an individual secret key

2. Git commit 564470e07d816f788d9c85acf72a1789c7787574

coefficient is non-negligible, since the two distributions are quite distinct. Note that the pdfs are less distinct than in Figure 7. This is because the bootstrapping error probability is less large and hence the rejection condition is less far away in the Gaussian tail.

To improve the accuracy of the guessing of the secret key, the KR$^{\mathsf{D}}$ attack averages all the failed ciphertext rounding errors. This amounts to averaging i.i.d. samples from the distributions of Figure 7. Thanks to the law of large numbers, the average behaves as a Gaussian random variable whose center depends on $s_i$. With a sufficiently large number of samples, the variance becomes sufficiently small to be able to accurately distinguish $s_i = 0$ from $s_i = 1$. In Figure 8, we plot the Hamming weight of $\mathbf{s} - \mathbf{s}_{\text{est}}$, as a function of the number of decryption failures, where $\mathbf{s}$ is the actual secret key and $\mathbf{s}_{\text{est}}$ is its guess. With 1 million samples, our attack recovered 597 secret key coefficients out of 600 for the custom parameter set. We also ran our KR$^{\mathsf{D}}$ attack on the TFHE-rs default parameters set using ciphertext samples generated by simulations. Our attack fully recovered the secret key with less than 256 decryption failures. One can query more samples to have an even better accuracy or perform an exhaustive search over secret keys that are close to the guess in Hamming distance.

## 6. About (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.

### 6.1. 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

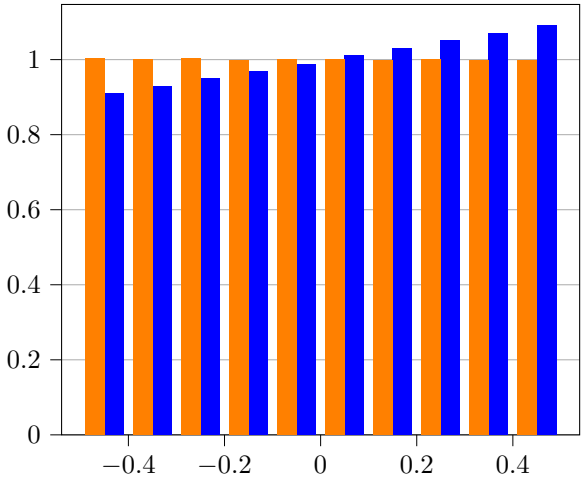



Figure 7: Experimental distributions of $\tilde{e}_i$ conditioned on decryption failure. Blue is for the $s_i = 1$ case, orange is for the $s_i = 0$ case.

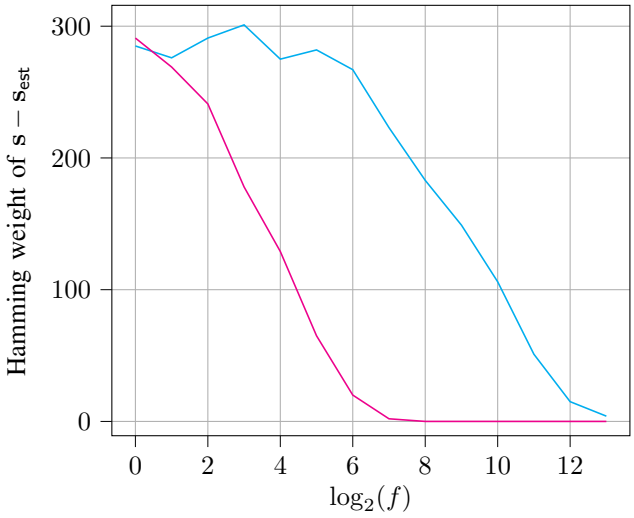



Figure 8: Performance of the attack as a function of the number of decryption failures. The cyan and magenta curves respectively correspond to the custom parameter set and the TFHE-rs default parameters.

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 |

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 $\mathsf{IND\text{-}CPA^D}$ 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 6.1** (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* $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PDec}, \mathsf{FinDec})$ *with the following specifications:*

- $\mathsf{Setup}$ *outputs secret keys* $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ *and a public key* $\mathsf{pk}$*;*
- $\mathsf{Enc}$ *takes as inputs a public key* $\mathsf{pk}$ *and a plaintext* $m \in \{0,1\}$*, and outputs a ciphertext* $\mathsf{ct}$*;*
- $\mathsf{Eval}$ *takes as inputs a public key* $\mathsf{pk}$*, 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* $\mathsf{ct}$*;*
- $\mathsf{PDec}$ *takes as inputs a secret key* $\mathsf{sk}_i$ *for* $i \leq n$*, and a ciphertext* $\mathsf{ct}$*, and outputs a partial decryption* $\mathsf{p}_i$*;*
- $\mathsf{FinDec}$ *takes as inputs a public key* $\mathsf{pk}$*, 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 $(\varepsilon, B)$-correct if for any key set* $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{pk})$ *output by* $\mathsf{Setup}$*, 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* $\mathsf{p}_i = \mathsf{PDec}_{\mathsf{sk}_i}(\mathsf{Eval}_{\mathsf{pk}}(C, (\mathsf{ct}_1, \ldots, \mathsf{ct}_k))$*:*

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

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

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

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

$\mathsf{Thres\text{-}IND\text{-}CPA}$ security is defined very similarly to $\mathsf{IND\text{-}CPA^D}$ security. Actually, $\mathsf{IND\text{-}CPA^D}$ security can be seen as $\mathsf{Thres\text{-}IND\text{-}CPA}$ security seeing the FHE scheme as a $(1,1)$-Threshold FHE scheme. We provide a detailed definition below. (This is an indistinguishability-based security definition, note that sometimes simulation-based definitions are preferred.)

**Definition 6.2** ($\mathsf{Thres\text{-}IND\text{-}CPA}$ 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 $\mathcal{O}_{\mathsf{Enc}}$, $\mathcal{O}_{\mathsf{Eval}}$ and $\mathcal{O}_{\mathsf{PDec}}$ defined below, we define the advantage* $\mathsf{Adv}^{\mathsf{Thres\text{-}IND\text{-}CPA}}(\mathcal{A})$ *of $\mathcal{A}$ against the* $\mathsf{Thres\text{-}IND\text{-}CPA}$ *security of $\Pi$ as:*

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

*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 $\mathcal{O}_{\mathsf{Enc}}$ and $\mathcal{O}_{\mathsf{Eval}}$ are defined exactly as for* $\mathsf{IND\text{-}CPA^D}$ *security. Oracle $\mathcal{O}_{\mathsf{PDec}}$ can be invoked only on a priorly obtained ciphertext. On input the index $j$ of a priorly generated ciphertext, oracle $\mathcal{O}_{\mathsf{PDec}}$ 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 6.3.** *Let $\Pi$ be a Threshold-FHE scheme and $\Pi^*$ be the underlying FHE scheme of $\Pi$. Let $\mathcal{B}$ an adversary against the* $\mathsf{IND\text{-}CPA^D}$ *security of $\Pi^*$. Then, there exists an adversary $\mathcal{A}$ against the* $\mathsf{Thres\text{-}IND\text{-}CPA}$ *security of $\Pi$, with same advantage and running time as $\mathcal{B}$.*

*Proof.* Let $\mathcal{B}$ be an adversary against the $\mathsf{IND\text{-}CPA^D}$ security of $\Pi^*$. We construct an adversary $\mathcal{A}$ against the security of $\Pi$. Adversary $\mathcal{A}$ obtains a public key $\mathsf{pk}$ 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 $\varnothing$). 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 $\mathsf{FinDec}$ on input the partial decryption shares it obtains from $\mathcal{O}_{\mathsf{PDec}}$, 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 $\mathsf{IND\text{-}CPA^D}$ challenger in $\mathcal{B}$'s view, hence leading to our claim. □

### 6.2. Noah's Ark, a Threshold-FHE Scheme

We first recall the Threshold-FHE scheme from [16]. It builds upon the CGGI FHE scheme, and thus the underlying FHE scheme is very similar to the one we studied in Section 5.

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 $Q$.

### 6.3. 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 [16], 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 Figure 6. For the sake of simplicity, we describe it for $n = t = 2$, even though this is not a parametrization considered in [16].

We recall in the columns of Table 1 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 [16].

## 7. Conclusion

We exhibited IND-CPA$^{\mathsf{D}}$ and KR$^{\mathsf{D}}$ 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 IND-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 IND-CPA$^{\mathsf{D}}$ attacks. This may lead to a performance penalty: for example, in the case of CKKS, a solution was given in [5] 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 [11].

**Algorithm 6** Attack on Noah's Ark, with $n = t = 2$.

$f := 0$
**for** $0 \le k < \gamma$ **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 $\mathsf{ct}'_k$
    $f := f + 1$
  **end if**
**end for**
Compute $\tilde{\mathbf{e}} = \frac{1}{f} \sum_{j=0}^{f-1} \tilde{\mathbf{e}}^j$
For all $i \le n$, compute $\tilde{s}_i = \begin{cases} 1 & \text{if } \tilde{\mathbf{e}}_i < \alpha/2 \\ 0 & \text{otherwise} \end{cases}$
**return** $\mathbf{s}_{\mathsf{est}}$

| $\rho$ | 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 |
| $\sigma_{\mathsf{ms}}$ | $2^{-8.49}$ | $2^{-9.41}$ | $2^{-8.29}$ | $2^{-9.29}$ |
| $2^{-\rho-1-1}$ | $2^{-3}$ | $2^{-6}$ | $2^{-3}$ | $2^{-6}$ |
| $r$ | $2^{5.49}$ | $2^{3.41}$ | $2^{5.29}$ | $2^{3.29}$ |
| $\mathrm{erfc}(r/\sqrt{2})$ | $2^{-1460}$ | $2^{-85.1}$ | $2^{-1110}$ | $2^{-72.8}$ |

TABLE 1: Parameter sets of Switch-and-Squash and lower bounds of decryption failure probability. The notations are borrowed from [16].

## References


[1] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. Fitzek, and N. Aaraj, "Survey on fully homomorphic encryption, theory, and applications," Cryptology ePrint Archive, Report 2022/1602, 2022, https://eprint.iacr.org/2022/1602.

[2] B. Li and D. Micciancio, "On the security of homomorphic encryption on approximate numbers," in *EUROCRYPT 2021, Part I*, ser. LNCS, A. Canteaut and F.-X. Standaert, Eds., vol. 12696. Springer, Heidelberg, Oct. 2021, pp. 648–677.

[3] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *ASIACRYPT 2017, Part I*, ser. LNCS, T. Takagi and T. Peyrin, Eds., vol. 10624. Springer, Heidelberg, Dec. 2017, pp. 409–437.

[4] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, pp. 34:1–34:40, 2009.

[5] B. Li, D. Micciancio, M. Schultz, and J. Sorrell, "Securing approximate homomorphic encryption using differential privacy," in *CRYPTO 2022, Part I*, ser. LNCS, Y. Dodis and T. Shrimpton, Eds., vol. 13507. Springer, Heidelberg, Aug. 2022, pp. 560–589.

[6] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, S. Goldwasser, Ed. ACM, 2012, pp. 309–325.

[7] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *CRYPTO 2012*, ser. LNCS, R. Safavi-Naini and R. Canetti, Eds., vol. 7417. Springer, Heidelberg, Aug. 2012, pp. 868–886.

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

[9] L. Ducas and D. Micciancio, "FHEW: Bootstrapping homomorphic encryption in less than a second," in *EUROCRYPT 2015, Part I*, ser. LNCS, E. Oswald and M. Fischlin, Eds., vol. 9056. Springer, Heidelberg, Apr. 2015, pp. 617–640.

[10] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," in *ASIACRYPT 2016, Part I*, ser. LNCS, J. H. Cheon and T. Takagi, Eds., vol. 10031. Springer, Heidelberg, Dec. 2016, pp. 3–33.

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

[12] 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 *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Los Angeles, CA, USA, 7 November 2022*, M. Brenner, A. Costache, and K. Rohloff, Eds. ACM, 2022, pp. 53–63, library available at https://www.openfhe.org/.

[13] ZAMA, "TFHE-rs v0.4," 2023, https://docs.zama.ai/tfhe-rs. [Online]. Available: https://docs.zama.ai/tfhe-rs

[14] 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 *EUROCRYPT 2012*, ser. LNCS, D. Pointcheval and T. Johansson, Eds., vol. 7237. Springer, Heidelberg, Apr. 2012, pp. 483–501.

[15] 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 *CRYPTO 2018, Part I*, ser. LNCS, H. Shacham and A. Boldyreva, Eds., vol. 10991. Springer, Heidelberg, Aug. 2018, pp. 565–596.

[16] 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 *Proceedings of the 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Copenhagen, Denmark, 26 November 2023*, M. Brenner, A. Costache, and K. Rohloff, Eds. ACM, 2023, pp. 35–46.

[17] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa, "Efficient public key encryption based on ideal lattices," in *ASIACRYPT 2009*, ser. LNCS, M. Matsui, Ed., vol. 5912. Springer, Heidelberg, Dec. 2009, pp. 617–635.

[18] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *EUROCRYPT 2010*, ser. LNCS, H. Gilbert, Ed., vol. 6110. Springer, Heidelberg, May / Jun. 2010, pp. 1–23.

[19] Q. Guo, D. Nabokov, E. Suvanto, and T. Johansson, "Key recovery attack on approximate homomorphic encryption with non-worst-case noise flooding countermeasures," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, 2024. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/guo

[20] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Des. Codes Cryptogr.*, vol. 75, no. 3, pp. 565–599, 2015.

[21] D. Dachman-Soled, L. Ducas, H. Gong, and M. Rossi, "LWE with side information: Attacks and concrete security estimation," in *CRYPTO 2020, Part II*, ser. LNCS, D. Micciancio and T. Ristenpart, Eds., vol. 12171. Springer, Heidelberg, Aug. 2020, pp. 329–358.

[22] D. Dachman-Soled, H. Gong, T. Hanson, and H. Kippen, "Revisiting security estimation for LWE with hints from a geometric perspective," in *CRYPTO 2023, Part V*, ser. LNCS, H. Handschuh and A. Lysyanskaya, Eds., vol. 14085. Springer, Heidelberg, Aug. 2023, pp. 748–781.

[23] 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.

[24] B. Biasioli, C. Marcolla, M. Calderini, and J. Mono, "Improving and automating bfv parameters selection: An average-case approach," Cryptology ePrint Archive, Paper 2023/600, 2023, https://eprint.iacr.org/2023/600. [Online]. Available: https://eprint.iacr.org/2023/600

[25] D. Micciancio and Y. Polyakov, "Bootstrapping in FHEW-like cryptosystems," IACR Cryptol. ePrint Arch., 2020, version dated October 23, 2022. [Online]. Available: https://eprint.iacr.org/2020/086

[26] M. V. Beirendonck, J. D'Anvers, F. Turan, and I. Verbauwhede, "FPT: A fixed-point accelerator for torus fully homomorphic encryption," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, W. Meng, C. D. Jensen, C. Cremers, and E. Kirda, Eds. ACM, 2023, pp. 741–755.

[27] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *J. Math. Cryptol.*, vol. 9, no. 3, pp. 169–203, 2015, software available at https://github.com/malb/lattice-estimator.