Drifting Towards Better Error Probabilities in Fully Homomorphic Encryption Schemes*

Olivier Bernard[1], Marc Joye[1], Nigel P. Smart[1,2], and Michael Walter[1]

[1] Zama, Paris, France
[2] COSIC, KU Leuven, Leuven, Belgium
firstname.lastname@zama.ai

Abstract. There are two security notions for FHE schemes: the traditional IND-CPA notion and a more stringent IND-CPA^{D} notion. These notions are equivalent when FHE schemes are perfectly correct. However, for schemes with negligible failure probability, the FHE parameters required to achieve IND-CPA^{D} security can be much larger than those needed to obtain IND-CPA security. This paper uses the notion of *ciphertext drift* in order to understand the practical difference between IND-CPA and IND-CPA^{D} security in schemes such as FHEW, TFHE, and FINAL. This notion allows us to define a modulus switching operation (the main culprit for the difference in parameters) such that one does not require adapting IND-CPA cryptographic parameters to meet the IND-CPA^{D} security level. Further, the extra cost incurred by the new techniques has no noticeable performance impact in practical applications. The paper also formally defines a stronger version for IND-CPA^{D} security called sIND-CPA^{D}, which is proved to be strictly separated from the IND-CPA^{D} notion. Criterion for turning an IND-CPA^{D} secure public-key encryption scheme into an sIND-CPA^{D} one is also provided.

Keywords: Fully homomorphic encryption · IND-CPA^{D} security · Modulus switching · Ciphertext drift · Noise analysis · Implementation

1 Introduction

The last fifteen years have seen rapid advances in the field of fully homomorphic encryption The initial work of Gentry [22] was truly groundbreaking in that it established not only (what we now call) a compact somewhat homomorphic encryption (SHE) scheme based on lattices, but it also presented a method to bootstrap the compact SHE scheme into a fully homomorphic encryption (FHE) scheme. Gentry's original scheme was based on properties of lattices of ideals of algebraic number fields, which are now considered insecure, but in the intervening years numerous authors have presented FHE schemes based on LWE [10], ring-LWE [11], NTRU [29], and the approximate integer GCD problem [37].

FHE and Failure Probability: FHE algorithms are much like normal encryption schemes, in that there is a set of three algorithms: a key generation algorithm, an encryption algorithm, and a decryption algorithm. The only difference lies in an additional evaluation algorithm which operates on ciphertexts. For example, given any two ciphertexts c and c' respectively encrypting messages m and m', there exists a public operation $\boxplus$ such that $c'' = c \boxplus c'$ is an encryption of $m'' = m + m'$. Being a *fully* homomorphic encryption scheme means that arbitrary functions can be applied to the ciphertexts indefinitely. However, the correctness of the scheme, i.e., the probability that

$$m = \mathrm{Dec}_{\mathsf{sk}}\Big(\mathrm{Eval}_{\mathsf{ek}}\big(g, (c_1, \ldots, c_k)\big)\Big)$$

* © IACR 2025. This paper is an extended version of the article published in the Proceedings of EUROCRYPT 2025, Part VIII, LNCS 15608, Springer, see https://doi.org/10.1007/978-3-031-91101-9_7.

where $m = g(m_1, \ldots, m_k)$ and, for all $i \in \{1, \ldots, k\}$, $m_i = \mathrm{Dec}_{\mathsf{sk}}(c_i)$, is not necessarily equal to one. The probability that the decryption algorithm returns an incorrect message is called the *failure probability* and is denoted p_{err}. Strictly speaking this failure probability is a function both of the scheme and of g. But for "polynomial" sized functions g we would like the failure probability p_{err} to be sufficiently small.

Controlling the Noise: All practical and secure methodologies for fully homomorphic encryption rely on hard lattice problems, accordingly, the resulting ciphertexts must contain a certain level of noise to guarantee the security of the encryption. The problem is that computing homomorphically tends to increase the noise level in the ciphertext. As long as the noise is below a certain threshold, the ciphertext can be decrypted. However, if the noise grows too much, it may overflow on the data itself, rendering decryption impossible. To prevent this from happening, a special noise-reduction operation called bootstrapping can be applied to the ciphertext, effectively resetting the noise to a nominal level. Following Gentry's discovery, successive generations of FHE emerged, aiming mostly at controlling the noise growth in homomorphic computations and/or improving the bootstrapping. Two approaches in particular have emerged:

- *Bootstrapped schemes:* FHE schemes are devised with the main goal of reducing as much as possible the computing overhead induced by the bootstrapping. Examples of such schemes include FHEW [19], TFHE [17] and its programmable extension [18], and FINAL [7].
- *Leveled schemes:* FHE schemes are parametrized so that the circuit representing a given function can be evaluated homomorphically without resorting to the bootstrap operation. As homomorphic multiplication introduces the most noise, what typically matters is the multiplicative depth (or number of levels) of the circuit being evaluated, that is, the largest sequence of consecutive multiplications. A leveled FHE scheme therefore provisions a noise budget so as to support L levels of multiplications where L is the multiplicative depth of the circuit. Examples of such FHE schemes include BFV [8,20], BGV [9], and CKKS [16].

Beyond Semantic Security: More recent years have seen the need to refine the "standard" encryption security notions when used with FHE schemes. For example, the recent notion of IND-CPA^{D} [27] (or *indistinguishability under chosen-plaintext attacks with a decryption oracle*) strengthens the usual IND-CPA security notion for FHE schemes. In the IND-CPA^{D} model, the attacker has additional access to a decryption oracle—the attacker is however severely restricted to the type of queries it may make to this oracle. More specifically, only ciphertexts for which the corresponding plaintext is known to the attacker can be submitted to the decryption oracle. This includes

- honestly generated ciphertexts by the attacker, and
- ciphertexts resulting from the evaluation of a *chosen* circuit on input ciphertexts whose matching plaintexts are known to the attacker.

It is important to note in the IND-CPA^{D} model that the circuit being homomorphically evaluated can be chosen by an IND-CPA^{D} attacker.

Modulus Switching: The key operation affecting the failure probability, and the one which enables homomorphic operations itself, for the bootstrapped schemes, is that of modulus switching. *Modulus switching* [10] is a way to change the modulus defining the ciphertexts from q to q', via scaling and rounding. As explained in [9,14], modulus switching is an essential tool for noise management in leveled fully homomorphic encryption. It is also

an essential tool in bootstrapped fully homomorphic encryption, as it is a key step (for example) prior the blind rotation with the so-called AP method [2,19] (incl. automorphism-based variants as for example in [6,26]) or the GINX method [21,17]. In both cases (leveled and bootstrapped), the challenge is to preserve correctness, namely that the ciphertexts correctly decrypt.

Our Contributions: The operation of modulus switching while central for enabling homomorphic operations may invalidate the correctness of the decryption. Worse, it may also affect the security. For example, the authors of [13,15] demonstrate attack scenarios wherein IND-CPA secure FHE schemes are not IND-CPA^{D} secure in the presence of decryption failures. In particular, the authors of [15] even show that the knowledge of failing ciphertexts may be turned into a key recovery attack in the IND-CPA^{D} model. The attacks in this extended model apply to any FHE scheme, including to schemes in the first category above; i.e., the bootstrapped schemes typified by FHEW, TFHE, and FINAL. These attacks are effective if the failure probability of the underlying FHE scheme is too high. However, if the failure probability in such schemes is set very low (in order to obtain for example 128 bits of security) then the parameters become prohibitively large. As aforementioned, decryption failures may result from the extra drift noise following a modulus switching. The main contribution of this paper are new modulus-switching methods that better control the failure probability.

Remarkably, the new modulus switching operations we introduce do not necessarily require adapting IND-CPA cryptographic parameters to meet the IND-CPA^{D} security level. Further, the extra cost incurred by the new techniques is minimal and has an unnoticeable performance impact in practical applications. Thus, the methods we introduce enable lower failure probabilities to be deployed in FHE schemes; thus preventing IND-CPA^{D} attacks as well as enabling smaller parameters. Two generic approaches are described. For each approach, several implementations featuring different advantages and disadvantages are presented. The proposed countermeasures against large drift are extensively studied and analyzed. Numerical experiments with real measurements confirm the results and concrete parameters we obtain from our analyses. As an illustration, it turns out that one of our generic defenses typically doubles the strength of the failure probability. For example, a TFHE parameter set designed to offer a failure probability of $p_{\mathrm{err}} = 2^{-64}$ with the *regular* modulus switching operation actually enjoys a failure probability of 2^{-128} when using our new modulus switching techniques, at an unnoticeable extra cost.

The second main contribution of this paper is definitional, noting that FHE schemes are seemingly in a security dilemma. On the one hand, it is generally expected that public-key cryptosystems are secure against chosen-ciphertext attacks in their practical deployment [35]. On the other hand, it is well known that FHE cannot meet such strong security requirements. The dilemma may be resolved by either embedding IND-CPA secure FHE schemes into protocols that ensure security even against active attackers [36] or to strengthen the scheme using additional cryptographic machinery [31]. When attempting to extend these approaches to approximate schemes or schemes with non-zero failure probability, we remark in this paper that the IND-CPA^{D} model may actually not be strong enough, at least in the public-key setting. The reason is a mismatch between how "honestly generated ciphertexts" are modeled in the IND-CPA^{D} model and how this is enforced in applications. In particular, in the corresponding applications the adversary may construct ciphertexts using the public key (which is the entire point of using a public-key scheme) and is then required to prove that the ciphertext is well-formed. The mismatch is that in the IND-CPA^{D} model, the adversary needs to submit the plaintexts to the encryp-

tion oracle, where it does not know or get to choose the randomness, while in applications it may choose the randomness as long as the resulting ciphertexts are still well-formed.

To address this issue, we provide a stronger notion of IND-CPA^{D} security and clarify its relationship to the IND-CPA^{D} definition. Perhaps surprisingly, we show that they are not equivalent, even for public-key schemes. The new security notion precisely captures how the failure probability needs to be computed in order to claim security in aforementioned applications. We show possible avenues to achieving it, including an outline of how to use our new modulus switching methods to improve security in this extended model, using TFHE as an example.

2 Preliminaries

Notations: For a positive integer q, we write $\mathbb{Z}/q\mathbb{Z}$ the ring of integers modulo q, which is identified with $[\![-\lfloor \frac{q}{2} \rfloor, \lceil \frac{q}{2} \rceil - 1]\!] \triangleq \{-\lfloor q/2 \rfloor, \ldots, 0, \ldots, \lceil q/2 \rceil - 1\}$. The operator $\lfloor \cdot \rceil$ denotes the rounding to the nearest integer, rounding downwards in the case of a tie. If $\mathcal{D}$ is a probability distribution, $a \leftarrow \mathcal{D}$ indicates that a is sampled according to $\mathcal{D}$. The uniform distribution over a set S is written $\mathcal{U}(S)$; the notation $a \xleftarrow{\$} S$ means that a is taken uniformly at random in S. The cardinality of a set S is denoted by $\sharp S$.

2.1 Probability and Statistics

It is useful to review a few basic concepts of probability theory and statistics. For a random variable X, its expected value, or *mean*, is denoted by $\mathbb{E}[X]$ and its *variance*, i.e., the expected value of the squared deviation from the mean, by $\mathrm{Var}(X)$.

The normal distribution with mean μ and variance σ^2 is written $\mathcal{N}(\mu, \sigma^2)$, and the *probability density function* of a random variable $N \sim \mathcal{N}\big(\mu, \sigma^2\big)$ is given by

$$\varphi_N(t) = \tfrac{1}{\sigma \cdot \sqrt{2\pi}} \cdot \exp\Big(-\tfrac{(t-\mu)^2}{2\sigma^2}\Big) \ .$$

The complementary error function, denoted by erfc, relates to the probability that a random variable $N \sim \mathcal{N}\big(\mu, \sigma^2\big)$ does *not* lie in $[\mu - r\sigma, \mu + r\sigma]$ for some parameter $r \geq 0$. More precisely, letting $p = \Pr\big[N \in [\mu - r\sigma, \mu + r\sigma]\big]$, one has

$$\Pr\big[N \notin [\mu - r\sigma, \mu + r\sigma]\big] = 1 - p = \mathrm{erfc}\big(\tfrac{r}{\sqrt{2}}\big) \ ,$$

which gives, if Φ denotes the cumulative distribution function of $\mathcal{N}(0,1)$, that

$$\Pr\big[N \leq x\big] = \int_{-\infty}^{x} \varphi_N(t)\,\mathrm{d}t = \tfrac{1}{2}\,\mathrm{erfc}\Big(-\tfrac{x-\mu}{\sigma\sqrt{2}}\Big) \triangleq \Phi\Big(\tfrac{x-\mu}{\sigma}\Big) \ . \tag{2.1}$$

The following table lists a number of values for $1 - p$ as a function of r.

r	1	2	3	...	7.15	9.16	10.29	13.11	16.13
$1-p$	31.73%	4.55%	0.27%	...	2^{-40}	2^{-64}	2^{-80}	2^{-128}	2^{-192}

2.2 Fully Homomorphic Encryption

A public-key *fully homomorphic encryption* (FHE) scheme is given by a tuple of algorithms $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec}, \mathrm{Eval})$ with the following properties:

– $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}) \leftarrow \mathrm{Gen}(1^\lambda)$. On input of the security parameter λ this probabilistic algorithm outputs the public key encryption key pk, the evaluation key ek, and the secret decryption key sk.
– $c \leftarrow \mathrm{Enc}_{\mathsf{pk}}(m)$. On input of the public key pk and a message $m \in \mathcal{M}$ in the message space, this probabilistic algorithm outputs a ciphertext c.
– $m \leftarrow \mathrm{Dec}_{\mathsf{sk}}(c)$. On input of a (valid) ciphertext c and the secret key sk this returns a message m.
– $c \leftarrow \mathrm{Eval}_{\mathsf{ek}}\big(g, (c_1, \ldots, c_k)\big)$. On input of the evaluation key ek, a function $g\colon \mathcal{M}^k \to \mathcal{M}$ and a sequence of k ciphertexts $(c_1, \ldots, c_k)$ this returns a ciphertext c.

Correctness: We not only require that $\mathrm{Dec}_{\mathsf{sk}}\big(\mathrm{Enc}_{\mathsf{pk}}(m)\big) = m$, but also that the Eval function returns with overwhelming probability ciphertexts which encrypt the message obtained by applying g to the messages encrypted by the input ciphertexts, i.e., if $m = g(m_1, \ldots, m_k)$ and, for all $i \in \{1, \ldots, k\}$, $m_i = \mathrm{Dec}_{\mathsf{sk}}(c_i)$,

$$m = \mathrm{Dec}_{\mathsf{sk}}\Big(\mathrm{Eval}_{\mathsf{ek}}\big(g, (c_1, \ldots, c_k)\big)\Big) \ .$$

IND-CPA^{D} Security: It has been established that IND-CPA security is not necessarily sufficient to secure certain FHE applications. Thus the more modern notion, see e.g., [27,28], is to consider a related notion called IND-CPA^{D} security.

Formally, IND-CPA^{D} considers the indistinguishability experiment IndExp given in Figure 2.1, for security parameter λ. The underlying security experiment is indexed by a random bit $b \in \{0, 1\}$. A common state S is maintained, which is made up of triplets of the form (m_0, m_1, c). The components of the jth entry of S are respectively accessed as $S[j].m_0$, $S[j].m_1$, and $S[j].c$.

$\mathrm{IndExp}_b(\lambda)$

1. The key generation algorithm is run to obtain keys pk, ek, and sk; $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}) \leftarrow \mathrm{Gen}(1^\lambda)$. Let $\mathcal{M}$ denote the message space.
2. The adversary $\mathcal{A}$ receives $(\mathsf{pk}, \mathsf{ek})$ and is given access to three oracles sharing a common state S initialized to $\emptyset$:
   – An encryption oracle Enc that on input a pair of messages $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$ returns the ciphertext $c \leftarrow \mathrm{Enc}_{\mathsf{pk}}(m_b)$. The state is updated as $S \leftarrow S \cup (m_0, m_1, c)$.
   – An evaluation oracle Eval that on input a function $g\colon \mathcal{M}^k \to \mathcal{M}$ and a sequence of k indexes $(j_1, \ldots, j_k) \in \{1, \ldots, \sharp S\}^k$ returns the ciphertext $c \leftarrow \mathrm{Eval}_{\mathsf{ek}}(g, S[j_1].c, \ldots, S[j_k].c)$. The state is updated as
   $$S \leftarrow S \cup \big(g(S[j_1].m_0, \ldots, S[j_k].m_0), g(S[j_1].m_1, \ldots, S[j_k].m_1), c\big) \ .$$
   – A decryption oracle Dec that on input an index $j \in \{1, \ldots, \sharp S\}$ checks whether $S[j].m_0 = S[j].m_1$ and, if so, returns $\mathrm{Dec}_{\mathsf{sk}}(S[j].c)$.
3. The adversary $\mathcal{A}$ interacts with the oracles and eventually outputs a bit b'.
4. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

Fig. 2.1: The IND-CPA^{D} security experiment $\mathrm{IndExp}_b(\lambda)$.

Informally, an FHE scheme is *perfectly* secure for the IND-CPA^{D} security notion if the best an attacker can do is to guess the value of b at random in the above experiment (i.e., the attacker outputs a random bit b'). With such a strategy, the attacker will recover the

value of b with probability $\frac{1}{2}$. The success probability is therefore defined as the distance between the probability that the guess $b' = b$ and $\frac{1}{2}$. This is formalized in Definition 2.1.

Definition 2.1. *A public-key FHE scheme is* IND-CPA^{D}-secure *if for any PPT adversary $\mathcal{A}$, it holds that*

$$\left|\Pr\left[\mathrm{IndExp}_b(\lambda) = 1\right] - \tfrac{1}{2}\right|$$

is negligible in security parameter λ.

3 Strengthening IND-CPA^{D} Security

The IND-CPA^{D} security model, as defined in Figure 2.1, captures the scenario where a user can only submit honestly generated (or evaluated) ciphertexts for decryption. In particular, the model implicitly assumes that, on input a k-variate function g and k indexes $j_1, \ldots, j_k$, the evaluation oracle returns a *valid* encryption of $g(S[j_1].m_b, \ldots, S[j_k].m_b)$—or at least, *with high probability*. Such an encryption scheme is termed *statistically correct* in [28].

A different scenario, considered in [13], where a user multiplies a ciphertext by a large scalar in order to have the noise overflowing the message, will *not* be considered as an attack in our model. In order to be considered within the security model, and therefore for the resulting ciphertext being qualified as 'honest', the corresponding scalar multiplication circuit (modeling function g) should apply bootstrappings when needed to contain the noise growth.

In this section we formalize and discuss a strengthened version of IND-CPA^{D} security, which we call *strong* IND-CPA^{D}or sIND-CPA^{D}.

3.1 Strong IND-CPA^{D}

We argue that the definition of IND-CPA^{D} security given in [27] (i.e., Definition 2.1) is too weak for certain practical applications, at least in the public key setting. The reason is that in the corresponding security game, an adversary can only submit fresh ciphertexts to the state S that have been generated using the encryption oracle, which generates *the randomness itself*, following the specifications of the scheme. This is very hard, if not impossible, to enforce in applications where the adversary may generate ciphertexts: even if the adversary is forced to prove the well-formedness of the ciphertexts, as, for example, in the protocol in [36] or, implicitly, in the construction in [12], this does not prove that the encryption randomness was indeed chosen according to the prescribed distribution. In order to fix this, we need the sIND-CPA^{D} notion.

As formalized in Figure 3.1, the sIND-CPA^{D} model is obtained by modifying the security experiment from Figure 2.1 by giving the adversary access to another encryption oracle Enc$'$. On input of a message $m \in \mathcal{M}$ and encryption randomness r, this oracle returns the ciphertext $c \leftarrow \mathrm{Enc}_{\mathsf{pk}}(m; r)$, and updates the state by setting $S \leftarrow S \cup (m, m, c)$. Note that we cannot simply allow the adversary to submit the randomness for the Left-or-Right-type oracle Enc, since that would render the definition unachievable.

Definition 3.1. *A public-key FHE scheme is sIND-CPA^{D}-secure if for any PPT adversary $\mathcal{A}$, it holds that*

$$\left|\Pr\left[\mathrm{sIndExp}_b(\lambda) = 1\right] - \tfrac{1}{2}\right|$$

is negligible in security parameter λ.

$\mathrm{sIndExp}_b(\lambda)$

1. The key generation algorithm is run to obtain keys pk, ek, and sk; $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}) \leftarrow \mathrm{Gen}(1^\lambda)$. The message space is denoted by $\mathcal{M}$ and the space of encryption randomness by $\mathcal{R}$.
2. The adversary $\mathcal{A}$ receives $(\mathsf{pk}, \mathsf{ek})$ and is given access to three oracles sharing a common state S initialized to $\emptyset$:
   - An encryption oracle Enc that on input a pair of messages $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$ returns the ciphertext $c \leftarrow \mathrm{Enc}_{\mathsf{pk}}(m_b)$. The state is updated as $S \leftarrow S \cup (m_0, m_1, c)$.
   - Another encryption oracle Enc′ that on input a message and randomness $(m, r) \in \mathcal{M} \times \mathcal{R}$ returns the ciphertext $c \leftarrow \mathrm{Enc}_{\mathsf{pk}}(m; r)$. The state is updated as $S \leftarrow S \cup (m, m, c)$.
   - An evaluation oracle Eval that on input a function $g\colon \mathcal{M}^k \to \mathcal{M}$ and a sequence of k indexes $(j_1, \ldots, j_k) \in \{1, \ldots, \sharp S\}^k$ returns the ciphertext $c \leftarrow \mathrm{Eval}_{\mathsf{ek}}(g, S[j_1].c, \ldots, S[j_k].c)$. The state is updated as
   $$S \leftarrow S \cup \big(g(S[j_1].m_0, \ldots, S[j_k].m_0), g(S[j_1].m_1, \ldots, S[j_k].m_1), c\big) \ .$$
   - A decryption oracle Dec that on input an index $j \in \{1, \ldots, \sharp S\}$ checks whether $S[j].m_0 = S[j].m_1$ and, if so, returns $\mathrm{Dec}_{\mathsf{sk}}(S[j].c)$.
3. The adversary $\mathcal{A}$ interacts with the oracles and eventually outputs a bit b'.
4. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

Fig. 3.1: The sIND-CPA^{D} security experiment $\mathrm{sIndExp}_b(\lambda)$.

Interestingly, the idea of taking randomness as an additional input to the encryption oracle already appears in [12] for proving security of one of their constructions; cf. Section 6.2 of [12]. However, the notion was not formally defined and the relationship to the IND-CPA^{D} model remains unclear. Below we show that the two notions are not equivalent and explore avenues to generically transform an IND-CPA^{D} secure scheme into an sIND-CPA^{D} secure one.

3.2 Separating IND-CPA^{D} and sIND-CPA^{D}

It is clear that sIND-CPA^{D} implies IND-CPA^{D}, since the oracles available to the adversary in the IND-CPA^{D} game are a strict subset of the ones in the sIND-CPA^{D} game. We now show that IND-CPA^{D} does not imply sIND-CPA^{D}, which means that sIND-CPA^{D} is strictly stronger than IND-CPA^{D}.

The two notions are very easily separated when considering symmetric-key schemes. For example, consider Regev-type encryption. If parameters are chosen properly, the scheme is IND-CPA^{D} secure. However, in the sIND-CPA^{D} model, choosing the encryption randomness of a ciphertext implies that the randomness is known; in particular, for Regev-type encryption, that means that the adversary knows the noise e and can remove it. This yields a linear equation in the secret key, which can be recovered after n queries where n denotes the LWE dimension.

The issue with symmetric-key encryption in the sIND-CPA^{D} model is that a ciphertext with known randomness may leak information about the secret key. This issue does not arise in public-key encryption, since here the adversary may construct the ciphertext itself using the public key. Accordingly, the new oracle does not provide the adversary with any additional information. So one might wonder if IND-CPA^{D} and sIND-CPA^{D} are equivalent for public-key schemes. We now show that we can even separate the two notions in the public-key setting.

Consider an arbitrary IND-CPA^{D} secure public-key FHE scheme. For simplicity, assume its message space is $\mathbb{Z}_2$. Modify encryption such that it takes an additional string of

randomness r' of length λ and define:

$$\mathrm{Enc}'_{\mathsf{pk}}(m; r, r') = \begin{cases} \mathrm{Enc}_{\mathsf{pk}}(1-m; r) & \text{if } r' = 0^\lambda \\ \mathrm{Enc}_{\mathsf{pk}}(m; r) & \text{otherwise} \end{cases} .$$

The modified scheme is still IND-CPA^{D} secure, since the probability that any ciphertext with $r' = 0^\lambda$ is added to the global game state S is negligible. So in the IND-CPA^{D} game, the two schemes are statistically indistinguishable. In contrast, the game is not sIND-CPA^{D} secure, since an adversary can request an encryption of m with $r' = 0^\lambda$, which will be added to the game state S and decrypts to $1 - m$. Accordingly, the scheme is susceptible to the generic distinguishing attack described in Appendix A.1 in the sIND-CPA^{D} game.

The next natural question is how to achieve sIND-CPA^{D} security. Note that existence of an sIND-CPA^{D} secure symmetric FHE scheme is trivial as long as an sIND-CPA^{D} secure asymmetric FHE exist, since one can simply keep the public key secret and call it a symmetric scheme, see, e.g., [34]. So for the rest of this section, we focus on the public-key setting.

3.3 A New Definition of Correctness for FHE

At the core of IND-CPA^{D} and sIND-CPA^{D} security of schemes with failure probability is the precise definition of correctness. Correctness of an FHE scheme can be defined in (at least) two different ways: either it requires that decryption succeeds with probability 1 (*perfectly correct*) or with overwhelming probability (*statistically correct*). An example of the latter definition is given in [27]:

Definition 3.2. *A homomorphic encryption scheme* $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec}, \mathrm{Eval})$ *is* statistically correct *if for all keys* $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}) \in \mathrm{Gen}(1^\lambda)$*, for all circuits* $g\colon \mathcal{M}^\ell \to \mathcal{M}$ *and for all* $m_i \in \mathcal{M}, 1 \le i \le \ell$*, it holds that*

$$\Pr\Big[\mathrm{Dec}_{\mathsf{sk}}\big(\mathrm{Eval}_{\mathsf{ek}}\big(g, (c_i)_{i=1}^\ell\big)\big) \neq g\big((m_i)_{i=1}^\ell\big) \Big| c_i \leftarrow \mathrm{Enc}_{\mathsf{pk}}(m_i) \text{ for } 1 \le i \le \ell\Big]$$

is negligible.

Clearly, perfect correctness is stronger than statistical correctness. For the IND-CPA^{D} model it is sufficient that an IND-CPA secure scheme is statistically correct as any such scheme is IND-CPA^{D} secure as proven in [27]. In other words, IND-CPA security and statistical correctness imply IND-CPA^{D} security. This is in contrast to the sIND-CPA^{D} definition. To see this, notice that the construction in the previous paragraph, separating IND-CPA^{D} and sIND-CPA^{D}, yields a scheme that is IND-CPA secure (since it is IND-CPA^{D} secure) and statistically correct, yet it is completely sIND-CPA^{D} insecure. So while IND-CPA security and statistical correctness imply IND-CPA^{D} security, they do not imply sIND-CPA^{D} security.

On the other hand, an IND-CPA secure scheme that is perfectly correct is sIND-CPA^{D} secure, meaning that IND-CPA security combined with perfect correctness guarantees sIND-CPA^{D} security. We can adapt the definition of statistical correctness such that it is also sufficient for sIND-CPA^{D} security:

Definition 3.3 (ACER Correctness). *A homomorphic encryption scheme defined by* $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec}, \mathrm{Eval})$ *is* statistically correct under adversarially chosen encryption randomness (ACER) *if for all keys* $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}) \in \mathrm{Gen}(1^\lambda)$*, for all circuits* $g\colon \mathcal{M}^\ell \mapsto \mathcal{M}$ *and for*

all $(m_i, r_i) \in \mathcal{M} \times \mathcal{R}, 1 \leq i \leq \ell$ *(where* $\mathcal{R}$ *is the randomness space of* Enc*), it holds that* $\mathrm{Dec}_{\mathsf{sk}}\big(\mathrm{Enc}_{\mathsf{pk}}(m_i; r_i)\big) = m_i$ *and*

$$\Pr\Big[\mathrm{Dec}_{\mathsf{sk}}\big(\mathrm{Eval}_{\mathsf{ek}}\big(g, (c_i)_{i=1}^{\ell}\big)\big) \neq g\big((m_i)_{i=1}^{\ell}\big)\Big| c_i \leftarrow \mathrm{Enc}_{\mathsf{pk}}(m_i; r_i) \text{ for } 1 \leq i \leq \ell\Big]$$

is negligible.

Notice the probability in the above definition is *only* over the randomness of Eval, so for schemes with deterministic Eval it is equivalent to perfect correctness.

Proposition 3.4. *Any public-key FHE scheme that is IND-CPA secure and ACER correct is sIND-CPA^{D} secure.*

Proof. The proof is similar to the one demonstrating that for statistically correct schemes, IND-CPA security implies IND-CPA^{D} security [27], with the additional observation that queries to the Enc′ oracle in the sIND-CPA^{D} game can be simulated by performing the encryption using the public key. Decryption queries on such ciphertexts can be answered correctly by returning the corresponding message due to the perfect correctness of fresh ciphertexts. □

We mention that there are further subtleties involved when defining imperfect correctness for FHE, as pointed out in [1], but these are orthogonal to this work, so we do not go into detail here.

3.4 Achieving ACER Correctness

A tempting approach to achieve ACER correctness could be to apply a random oracle to the input randomness to obtain the encryption randomness in order to force the latter to be uniform. Unfortunately, that does not work as we show next. The reason is that the adversary may still bias the encryption randomness by using rejection sampling.

Consider a standard asymmetric LWE-based additively homomorphic scheme and restrict the supported circuit class to circuits with up to k additions. Assume parameters (in particular the noise distribution) are set up in a way that up to k additions are possible with failure probability p *assuming the noise is indeed sampled from the prescribed centered distribution* over $\mathbb{Z}$, where p is negligible. This means that the scheme is statistically correct. Apply the transformation that derives the randomness to sample the noise by applying a random oracle to an input string of randomness in the hope of achieving the same level of ACER correctness. An adversary can now construct k ciphertexts, where all of them have noise component with the same sign, simply by trial and error. In expectation this only requires $2k$ trial encryptions. Clearly, summing k such ciphertexts will have a higher failure probability $p' > p$, which may be non-negligible. It follows that the scheme is not ACER correct.

We now give a sufficient condition for statistically correct schemes that allows the construction of an ACER correct scheme. Specifically, if a statistically correct scheme has re-randomizable ciphertexts, then there is a simple transformation that achieves ACER correctness. The idea is to re-randomize each input ciphertext before running Eval. Combining this with Proposition 3.4 shows that if the scheme was IND-CPA^{D} secure before this transformation, the transformed scheme is sIND-CPA^{D} secure.

Proposition 3.5. *Let* $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec}, \mathrm{Eval})$ *be a public-key homomorphic encryption scheme with ciphertext space* $\mathcal{C}$ *and encryption randomness space* $\mathcal{R}$*. If the scheme is statistically correct,* $\Pr[\mathrm{Dec}_{\mathsf{sk}}(\mathrm{Enc}_{\mathsf{pk}}(m)) = m] = 1$*, and there exists a PPT algorithm*

re-rand: $\mathcal{C} \to \mathcal{C}$ *such that for any* $(m, r) \in \mathcal{M} \times \mathcal{R}$ *the distributions* $\{\text{re-rand}(\text{Enc}_{\mathsf{pk}}(m; r))\}$ *and* $\{\text{Enc}_{\mathsf{pk}}(m)\}$ *are statistically indistinguishable, then the scheme, where* Eval' *is defined as*

$$\text{Eval}'_{\mathsf{ek}}(g, c_1, \ldots, c_\ell) = \text{Eval}_{\mathsf{ek}}(g, \text{re-rand}(c_1), \ldots, \text{re-rand}(c_\ell)) ,$$

is ACER correct.

Proof. By the definition of statistical correctness, it holds that, for all keys $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk})$ in the support of $\text{Gen}(1^\lambda)$, for all circuits $g\colon \mathcal{M}^\ell \to \mathcal{M}$ and for all $m_i \in \mathcal{M}, 1 \le i \le \ell$, we have that

$$\Pr\left[\text{Dec}_{\mathsf{sk}}(\text{Eval}_{\mathsf{ek}}(g, (c_i)_{i=1}^\ell)) \neq g((m_i)_{i=1}^\ell) \middle| c_i \leftarrow \text{Enc}_{\mathsf{pk}}(m_i) \text{ for } 1 \le i \le \ell\right]$$

is negligible. By the property of re-rand, it follows that

$$\Pr\left[\text{Dec}_{\mathsf{sk}}(\text{Eval}_{\mathsf{ek}}(g, \text{re-rand}(c_i)_{i=1}^\ell)) \neq g((m_i)_{i=1}^\ell) \middle| c_i \leftarrow \text{Enc}_{\mathsf{pk}}(m_i; r), 1 \le i \le \ell\right]$$

is also negligible for all $(m_i, r) \in \mathcal{M} \times \mathcal{R}$. The proposition follows. □

4 Ciphertext Drift

Our main results are based on the concept of *ciphertext drift*. To understand ciphertext drift one has to dig into the modulus switching algorithm used in Regev-style [33] FHE encryption algorithms. To fix notation in Figure 4.1 we present a basic Regev-style encryption scheme (just the main algorithms and not any homomorphic operations). It should be noted that the correctness of the decryption requires the noise e present in a ciphertext $\boldsymbol{c}$ satisfies $|e| < \Delta/2$. Typically, the error distribution χ used in Regev-type is a discretized or discrete version of the normal distribution $\mathcal{N}(0, \sigma^2)$ with mean 0 and (small) variance σ^2.

Regev Encryption

Key generation On input security parameter, select positive integers n, t, q with t dividing q, let $\Delta = q/t$, and define a discretized error distribution χ over $\mathbb{Z}$. Finally, sample a random vector $\boldsymbol{s} = (s_1, \ldots, s_n) \xleftarrow{\$} \mathcal{U}(\{0,1\}^n)$.

Encryption The encryption of a message $m \in \mathbb{Z}/t\mathbb{Z}$ (viewed as an integer in $\{0, \ldots, t-1\}$) is given by $\boldsymbol{c} = (a_1, \ldots, a_n, b) \in (\mathbb{Z}/q\mathbb{Z})^{n+1}$ with

$$\begin{cases} (a_1, \ldots, a_n) \xleftarrow{\$} (\mathbb{Z}/q\mathbb{Z})^n \\ b = \sum_{j=1}^n a_j \cdot s_j + \mu + e \pmod q \end{cases}$$

and $\mu = \Delta \cdot m$, where $e \in \mathbb{Z}$ is a (small) noise drawn randomly from χ.

Decryption To decrypt $\boldsymbol{c} = (a_1, \ldots, a_n, b)$, using $\boldsymbol{s} = (s_1, \ldots, s_n)$, return $\lfloor \mu^*/\Delta \rceil \bmod t$ where $\mu^* = b - \sum_{j=1}^n a_j \cdot s_j \pmod q$.

Fig. 4.1: A simple Regev-style encryption scheme.

4.1 Modulus Switching and the Drift Vector/Error

In Figure 4.2 we define the modulus switching operation for Regev-style encryption. The modulus switching operation forms the key to the bootstrapped FHE algorithms such as FHEW, TFHE, and FINAL. It is also the place in these algorithms where the most

ModSwitch

With the previous notations for Regev encryption, a Regev ciphertext modulo q, i.e., $\boldsymbol{c} = \left(a_1, \ldots, a_n, b = \sum_{j=1}^{n} a_j \cdot s_j + \Delta \cdot m + e\right) \in (\mathbb{Z}/q\mathbb{Z})^{n+1}$, encrypting a message m, is converted to a Regev ciphertext modulo q'

$$\tilde{\boldsymbol{c}} = (\tilde{a}_1, \ldots, \tilde{a}_n, \tilde{b}) \in (\mathbb{Z}/q'\mathbb{Z})^{n+1}$$

where $\begin{cases} \tilde{a}_j = \left\lfloor \frac{a_j}{q} \cdot q' \right\rceil \pmod{q'} & \text{for } 1 \leq j \leq n \\ \tilde{b} = \left\lfloor \frac{b}{q} \cdot q' \right\rceil \pmod{q'} \end{cases}$.

Fig. 4.2: The modulus switching operation for Regev-style ciphertexts.

ciphertext noise is introduced into the data; and thus the place where the failure probability of the scheme is most acute.

Assume for simplicity that q' divides q in the modulus switching algorithm in Figure 4.2 (which is the most important case for practical instantiations of FHEW, TFHE, and FINAL). The *drift vector* is the vector consisting of the scaling/rounding errors resulting from modulo switching ciphertext components modulo q to modulo q'. The extra error term resulting from the modulus switching, that adds to the noise already present in the input ciphertext $\boldsymbol{c}$, is called the *drift noise* and is denoted e_{drift}. Hence, writing

$$\begin{cases} \alpha_j = \tilde{a}_j \cdot \frac{q}{q'} - a_j & \text{for some } \alpha_j \in \left[\!\left[-\frac{q}{2q'}, \frac{q}{2q'} \right]\!\right] \\ \beta = \tilde{b} \cdot \frac{q}{q'} - b & \text{for some } \beta \in \left[\!\left[-\frac{q}{2q'}, \frac{q}{2q'} \right]\!\right] \end{cases},$$

the drift vector is given by $(\alpha_1, \ldots, \alpha_n, \beta) \in \mathbb{Z}^{n+1}$ and the drift noise is given by

$$e_{\text{drift}} = \beta - \sum_{j=1}^{n} \alpha_j \cdot s_j \in \mathbb{Z} \ .$$

Indeed, letting $\Delta' = q'/t$, it can be verified that

$$\begin{aligned} \tfrac{q}{q'} \cdot \left[\left(\tilde{b} - \textstyle\sum_{j=1}^{n} \tilde{a}_j \cdot s_j - \Delta' \cdot m \right) \bmod q' \right] \\ &\equiv \beta + b - \textstyle\sum_{j=1}^{n} (\alpha_j + a_j) \cdot s_j - \Delta \cdot m \pmod{q} \\ &= e + e_{\text{drift}} \ . \end{aligned}$$

If $|e + e_{\text{drift}}| < \Delta/2$, then the mod-switched ciphertext $\tilde{\boldsymbol{c}}$ correctly decrypts to m.

4.2 Affect on FHE Operations: TFHE Case Study

To understand the affect of the drift on FHE operations we examine the implications for TFHE [17] in particular. For TFHE we have that q and t in Regev's scheme are both powers of two, with $q \gg t$. A TFHE ciphertext of a message $m \in \mathbb{Z}/t\mathbb{Z}$ under private key $\boldsymbol{s} \in \{0,1\}^n$ is a vector of the form

$$\left(\boldsymbol{a} = (a_1, \ldots, a_n), b = \textstyle\sum_{i=1}^{n} a_i \cdot s_i + \Delta \cdot m + e \right) \in (\mathbb{Z}/q\mathbb{Z})^{n+1}$$

for some noise error e satisfying $|e| < \frac{\Delta}{2}$. The *programmable bootstrapping* in TFHE enables to transform an encryption of m into an encryption of $f(m)$ for some function $f\colon \mathbb{Z}/t\mathbb{Z} \to$

$\mathbb{Z}/t\mathbb{Z}$; the regular bootstrapping corresponds the function f being the identity map. The salient feature of this operation is that it resets the noise present in the resulting ciphertext to a nominal value. We refer the reader to [23] for details.

In very rare cases, namely with failure probability p_{err}, it may happen that the programmable bootstrapping does not return a TFHE encryption of $f(m)$. The main source of error originates from the modulus switch occurring prior to the blind rotation.

In TFHE the modulus switching operation maps the modulus from the input ciphertext modulus q, to the output ciphertext modulus $q' = 2N$, which is also a power for two. The value N here coming from the dimension of a ring-LWE instance. In other words the input TFHE ciphertext is[3]

$$\boldsymbol{C} \leftarrow \left(\boldsymbol{a} = (a_1, \ldots, a_n), b = \textstyle\sum_{i=1}^{n} a_i \cdot s_i + \Delta \cdot m + e_{\text{in}}\right) \in (\mathbb{Z}/q\mathbb{Z})^{n+1}$$

whilst the output ciphertext is the vector $(\tilde{a}_1, \ldots, \tilde{a}_n, \tilde{b}) \in (\mathbb{Z}/2N\mathbb{Z})^{n+1}$ with

$$\begin{cases} \tilde{a}_i = \left\lfloor \frac{a_i}{q} \cdot 2N \right\rceil & \text{for } i \in \{1, \ldots, n\} \\ \tilde{b} = \left\lfloor \frac{b}{q} \cdot 2N \right\rceil \end{cases} .$$

The resulting ciphertext fails to decrypt correctly when

$$f\left(\left\lfloor \frac{\tilde{b} - \sum_{i=1}^{n} \tilde{a}_i \cdot s_i \pmod{2N}}{2N/t} \right\rceil \right) \neq f(m) \pmod{t} .$$

In order to cover every function f, including the identity map, we need to analyze

$$\left\lfloor \frac{\tilde{b} - \sum_{i=1}^{n} \tilde{a}_i \cdot s_i \pmod{2N}}{2N/t} \right\rceil \tag{4.1}$$

as an approximation of m.

As aforementioned, elements modulo q (resp. modulo $2N$) are represented as integers in $[\![-q/2, q/2 - 1]\!]$ (resp. $[\![-N, N-1]\!]$). Hence, as shown in Lemma 4.1, we can write $a_i = \tilde{a}_i \cdot \frac{q}{2N} - \alpha_i$ for some $\alpha_i \in [\![-\frac{q}{4N}, \frac{q}{4N} - 1]\!]$ and $b = \tilde{b} \cdot \frac{q}{2N} - \beta$ for some $\beta \in [\![-\frac{q}{4N}, \frac{q}{4N} - 1]\!]$. Thus, letting $Q' = q/(2N)$, we obtain

$$\begin{aligned} &\left(\tilde{b} - \textstyle\sum_{i=1}^{n} \tilde{a}_i \cdot s_i - \frac{2N}{t} \cdot m\right) \bmod 2N \\ &\qquad = \frac{Q'\left(\tilde{b} - \sum_{i=1}^{n} \tilde{a}_i \cdot s_i - \frac{2N}{t} \cdot m\right) \bmod q}{Q'} \\ &\qquad = \frac{\left((b+\beta) - \sum_{i=1}^{n} (a_i + \alpha_i) s_i - \Delta \cdot m\right) \bmod q}{Q'} \\ &\qquad = \frac{\left(e_{\text{in}} + \left(\beta - \sum_{i=1}^{n} \alpha_i \cdot s_i\right)\right) \bmod q}{Q'} . \end{aligned}$$

We observe that there are two error components in the above expression: $e_{\text{in}} \coloneqq \text{Err}(\boldsymbol{C})$ and $e_{\text{drift}} \coloneqq \beta - \sum_{i=1}^{n} \alpha_i \cdot s_i$. Analysing the error values e_{in} and e_{drift} form the backbone of this paper.

Write $q = 2^{\Omega}$, $N = 2^{\nu-1}$ and $Q' = \frac{q}{2N} = 2^{\Omega-\nu}$ where $\nu < \Omega$. The following lemma examines the conversion error when an element of $\mathbb{Z}/q\mathbb{Z}$ is rescaled as Q' times an element of $\mathbb{Z}/2N\mathbb{Z}$.

[3] For more clarity, the noise error present in the input ciphertext is denoted $e_{\text{in}} \in \mathbb{Z}$.

Lemma 4.1. *Let $k \in \mathbb{Z}/q\mathbb{Z}$. Then $\tilde{k} \coloneqq \left\lfloor \frac{k}{q} \cdot 2N \right\rceil \pmod{2N}$ satisfies $Q' \cdot \tilde{k} = k + \kappa$ with $\kappa \in [\![-\frac{q}{4N}, \frac{q}{4N} - 1]\!] \subseteq \mathbb{Z}/q\mathbb{Z}$. Furthermore, if $k \leftarrow \mathcal{U}(\mathbb{Z}/q\mathbb{Z})$ then $\tilde{k}$ is uniform over $\mathbb{Z}/2N\mathbb{Z}$ and κ is uniform over $[\![-\frac{q}{4N}, \frac{q}{4N} - 1]\!]$.*

Proof. There are two cases to consider.

1. $\boxed{4N = q:}$ Suppose that $k \in [\![-\frac{q}{2}, \frac{q}{2} - 1]\!]$ is even. Then $\left\lfloor \frac{k}{q} \cdot 2N \right\rceil = \frac{k}{2}$ and $\left\lfloor \frac{k+1}{q} \cdot 2N \right\rceil = \left\lfloor \frac{k}{2} + \frac{1}{2} \right\rceil = \frac{k}{2}$. We deduce that $\tilde{k} \in [\![\frac{-q/2}{2}, \frac{q/2-2}{2}]\!] = [\![-N, N-1]\!]$ and each value occurs exactly twice. We also deduce that $\kappa = Q' \cdot \left\lfloor \frac{k}{q} \cdot 2N \right\rceil - k = 2 \cdot \frac{1}{2} - k = 0$ or $\kappa = Q' \cdot \left\lfloor \frac{k+1}{q} \cdot 2N \right\rceil - (k+1) = 2 \cdot \frac{k}{2} - (k+1) = -1$.
2. $\boxed{4N < q:}$ We decompose the interval for k as $[\![-\frac{q}{2}, \frac{q}{2} - 1]\!] = [\![-\frac{q}{2}, \frac{q}{2} - \frac{q}{4N}]\!] \cup [\![\frac{q}{2} - \frac{q}{4N} + 1, \frac{q}{2} - 1]\!]$ so that

$$\left\lfloor \frac{k}{q} \cdot 2N \right\rceil \in \begin{cases} [\![-N, N-1]\!] & \text{if } k \in [\![-\frac{q}{2}, \frac{q}{2} - \frac{q}{4N}]\!] \\ \{N\} & \text{if } k \in [\![\frac{q}{2} - \frac{q}{4N} + 1, \frac{q}{2} - 1]\!] \end{cases} .$$

In the latter case, since $\tilde{k} = \left\lfloor 2N \cdot \frac{k}{q} \right\rceil \pmod{2N} \in \mathbb{Z}/2N\mathbb{Z}$, we have $\tilde{k} = -N$ when $k \in [\![\frac{q}{2} - \frac{q}{4N} + 1, \frac{q}{2} - 1]\!]$. Therefore, we observe that

$$\tilde{k} = \begin{cases} -N & \text{if } k \in [\![\frac{q}{2} - \frac{q}{4N} + 1, \frac{q}{2} - 1]\!] \cup [\![-\frac{q}{2}, -\frac{q}{2} + \frac{q}{4N}]\!] \\ -N+1 & \text{if } k \in [\![-\frac{q}{2} + \frac{q}{4N} + 1, -\frac{q}{2} + \frac{3q}{4N}]\!] \\ -N+2 & \text{if } k \in [\![-\frac{q}{2} + \frac{3q}{4N} + 1, -\frac{q}{2} + \frac{5q}{4N}]\!] \\ \quad \vdots & \\ N-1 & \text{if } k \in [\![-\frac{q}{2} + \frac{(4N-3)q}{4N} + 1, -\frac{q}{2} + \frac{(4N-1)q}{4N}]\!] \end{cases} .$$

It can be verified that each sub-case has a range of cardinality $q/(2N)$ for t. As a result, each possible value for $\tilde{k} \in [\![-N, N-1]\!]$ occurs exactly $q/(2N)$ times and, in turn, $Q' \cdot \tilde{k} - k \in \{\frac{q}{4N} - i \mid 1 \le i \le \frac{q}{2N}\}$. Letting $\mathcal{S} = \{\frac{q}{4N} - i \mid 1 \le i \le \frac{q}{2N}\}$, it can be rewritten as

$$\begin{aligned} \mathcal{S} &= \{\tfrac{q}{4N} - (-i + \tfrac{q}{2N}) \mid 1 \le -i + \tfrac{q}{2N} \le \tfrac{q}{2N}\} \\ &= \{-\tfrac{q}{4N} + i \mid 0 \le i \le \tfrac{q}{2N} - 1\} \\ &= [\![-\tfrac{q}{4N}, \tfrac{q}{4N} - 1]\!] . \end{aligned}$$

Consequently, in all cases, if $k \leftarrow \mathcal{U}(\mathbb{Z}/q\mathbb{Z})$ then $\tilde{k} \in \mathcal{U}(\mathbb{Z}/2N\mathbb{Z})$ and $\kappa = Q' \cdot \tilde{k} - k \in \mathcal{U}(\{\frac{q}{4N} - i \mid 1 \le i \le \frac{q}{2N}\})$. □

Let $\mathcal{S} = [\![-\frac{q}{4N}, \frac{q}{4N} - 1]\!]$. From Lemma 4.1, we have $Q' \cdot \tilde{a}_i = a_i + \alpha_j$ where $\alpha_i \in \mathcal{U}(\mathcal{S})$ and $Q' \cdot \tilde{b} = b + \beta$ where $\beta \in \mathcal{U}(\mathcal{S})$. Since α_j is uniform, it follows that $\mathbb{E}[\alpha_i] = \frac{4N}{2q} \sum_{j=-q/(4N)}^{q/(4N)-1} j = \frac{2N}{q} \cdot \frac{-q}{4N} = -\frac{1}{2}$ and, similarly, $\mathbb{E}[{\alpha_i}^2] = \frac{4N}{2q} \sum_{j=-q/(4N)}^{q/(4N)-1} j^2 = \frac{2N}{q} \left(\left(\frac{-q}{4N}\right)^2 + 2 \sum_{j=1}^{q/(4N)-1} j^2 \right) = \frac{2N}{q} \left(\frac{q^2}{16N^2} + 2 \frac{(\frac{q}{4N}-1)\frac{q}{4N}(2(\frac{q}{4N}-1)+1)}{6} \right) = \frac{q^2+8N^2}{48N^2}$. We so obtain $\mathrm{Var}(\alpha_i) = \mathbb{E}[{\alpha_i}^2] - \mathbb{E}[\alpha_i]^2 = \frac{q^2-4N^2}{48N^2}$. Likewise, under the LWE assumption, we have $\mathbb{E}[\beta] = -\frac{1}{2}$ and $\mathrm{Var}(\beta) = \frac{q^2-4N^2}{48N^2}$.

The drift error is defined as $e_{\text{drift}} = \beta - \sum_{i=1}^{n} \alpha_i \cdot s_i$. Noting that $\mathbb{E}[s_i] = \frac{1}{2}$ and $\mathrm{Var}(s_i) = \frac{1}{4}$ for binary keys, we get

$$\begin{aligned} \mathbb{E}[e_{\text{drift}}] &= \mathbb{E}[\beta] - n \cdot \mathbb{E}[\alpha_i \cdot s_i] = \mathbb{E}[\beta] - n \cdot E[\alpha_i] \cdot \mathbb{E}[s_i] \\ &= \frac{n-2}{4} \end{aligned} \tag{4.2}$$

and

$$\begin{aligned}\mathrm{Var}(e_{\mathrm{drift}}) &= \mathrm{Var}(\beta) + n\,\mathrm{Var}(s_i \cdot \alpha_i) \\ &= \mathrm{Var}(\beta) + n\big(\mathrm{Var}(s_i)\,\mathrm{Var}(\alpha_i) + \\ &\qquad\qquad \mathrm{Var}(s_i)\cdot\mathbb{E}[\alpha_i]^2 + \mathrm{Var}(\alpha_i)\cdot\mathbb{E}[s_i]^2\big) \\ &= \frac{(n+2)q^2}{96N^2} + \frac{n-4}{48}\ . \end{aligned} \tag{4.3}$$

Remark 4.2. For an even r, instead of representing elements of $\mathbb{Z}/r\mathbb{Z}$ as $[\![-\frac{r}{2}, \frac{r}{2}-1]\!]$, we could consider the balanced set $[\![-\frac{r}{2}, \frac{r}{2}]\!]$ and use indiscriminately $-\frac{r}{2}$ or $\frac{r}{2}$. The rounding operation would then round uniformly at random upwards or downwards in the case of a tie. The previous analysis can be adapted to this setting. With the previous notations, we obtain $\mathbb{E}[\alpha_i] = \mathbb{E}[\beta] = 0$ and $\mathrm{Var}(\alpha_i) = \mathrm{Var}(\beta) = \frac{q^2+8N^2}{48N^2}$, which leads to

$$\mathbb{E}[e_{\mathrm{drift}}] = 0 \quad \text{and} \quad \mathrm{Var}(e_{\mathrm{drift}}) = \frac{(n+2)q^2}{96N^2} + \frac{n+2}{12}\ .$$

In practice, this does not change much the variance since the term $\frac{(n+2)q^2}{96N^2}$ dominates (typical parameters include $N \in \{2^{10}, 2^{11}, 2^{12}\}$ and $q \in \{2^{32}, 2^{64}\}$).

5 Countermeasures Against Large Drift

In this section we outline two general countermeasures against large drift values for FHE schemes. Our countermeasures are based on two important observations:

1. A ciphertext that is produced by a partially or fully homomorphic encryption scheme can easily be transformed to another ciphertext encrypting the same plaintext, under the same scheme.
2. Modulus switching is a public operation, there are no secrets involved; in particular, the private key is not required. As a result, the drift vector, resulting from the modulus switching operation, can be publicly computed from the input ciphertext modulo q and the resulting mod-switched ciphertext modulo q'.

5.1 Ciphertext Transformation

To explain the first observation, note that if the encryption scheme is additively homomorphic, another ciphertext of the same plaintext may be obtained by adding an encryption of zero. Likewise, if the encryption scheme is multiplicatively homomorphic, one may instead multiply with an encryption of one. For the FHE schemes considered in this paper, one may either add an encryption of zero or multiply with an encryption of one to get another ciphertext encrypting the same plaintext.

In what follows we focus on addition as a way of transforming the ciphertext. We let $\boxplus$ (resp. $\cdot$) denote the ciphertext addition (resp. multiplication of a ciphertext by a scalar). We assume that a set of Z encryptions of zero, $\boldsymbol{D}_1, \ldots, \boldsymbol{D}_Z$, are provided as public parameters of the scheme (for example they are included in the evaluation key ek). The process of adding an encryption of zero to transform a given ciphertext can be deterministic, random, or pseudo-random.

Let $\boldsymbol{c}$ denote the input ciphertext to our transformation process, we will denote the transform of $\boldsymbol{c}$ to a new ciphertext encrypting the same value by

$$\boldsymbol{c}' \leftarrow \mathrm{Transform}\big(\boldsymbol{c}, (\boldsymbol{D}_1, \ldots, \boldsymbol{D}_Z), \mathsf{cnt}\big)\ ,$$

where cnt is a counter. Upon each call to Transform the counter cnt is incremented by one. We assume, for ease of exposition and syntactic convenience, that on the first call to Transform for an input ciphertext $\boldsymbol{c}$ the value of cnt is set to 1, and that Transform($\boldsymbol{c}$, $(\boldsymbol{D}_1, \ldots, \boldsymbol{D}_Z), \mathsf{cnt}$) with $\mathsf{cnt} = 0$ outputs the input ciphertext $\boldsymbol{c}$. The algorithm Transform can take one of the following forms, depending on the desired security and functional properties of the scheme:

- *Deterministic Transformation Process:* Here the input ciphertext $\boldsymbol{c}$ is transformed into $\boldsymbol{c} \boxplus \boldsymbol{D}_{\mathsf{cnt}}$, for $\mathsf{cnt} = 1, 2, \ldots, Z$. If further calls to transformations are needed, ciphertext can for example be transformed into $\boldsymbol{c} \boxplus \boldsymbol{D}_1 \boxplus \boldsymbol{D}_{\mathsf{cnt}-Z}$, for $\mathsf{cnt} = Z+1, \ldots, 2 \cdot Z$. And so on with $\boldsymbol{D}_2, \boldsymbol{D}_3, \ldots, \boldsymbol{D}_Z$; next with $\boldsymbol{D}_1 \boxplus \boldsymbol{D}_2, \boldsymbol{D}_1 \boxplus \boldsymbol{D}_3, \ldots, \boldsymbol{D}_1 \boxplus \boldsymbol{D}_Z$, and so on. There are numerous possible variants; what matters is to keep the deterministic nature of the process.
- *Random Transformation Process:* Here the input ciphertext $\boldsymbol{c}$ is transformed into $\boldsymbol{c} \boxplus \boldsymbol{D}_i$ where i is chosen at random in $\{1, \ldots, n\}$. More generally, the input ciphertext $\boldsymbol{c}$ is transformed by adding to it a random linear combination of $\boldsymbol{D}_1, \ldots, \boldsymbol{D}_Z$; specifically, $\boldsymbol{c}$ is replaced with $\boldsymbol{c} \boxplus \boldsymbol{R}$ where $\boldsymbol{R} = \theta_1 \cdot \boldsymbol{D}_1 \boxplus \cdots \boxplus \theta_Z \cdot \boldsymbol{D}_Z$ and $\theta_1, \ldots, \theta_Z$ are (small) random scalars.
- *Pseudo-random Transformation Process:* This can be seen as a specialization of the previous case where a pseudo-random linear combination of $\boldsymbol{D}_1, \ldots, \boldsymbol{D}_Z$ is taken (as opposed to a random one). This is can be obtained through applying a pseudo-random function to the input ciphertext, the value cnt, and (potentially) other public parameters.

In the case of a public-key FHE, the encryptions of zero can alternatively be obtained using the public encryption key and the encryption algorithm. Again, the process of obtaining such encryptions of zero can be deterministic, random, or pseudo-random depending on whether the random coins for the encryption process are determined deterministically, randomly, or pseudo-randomly.

5.2 Ciphertext Quality Test

The general idea behind all of our countermeasures is to transform an input ciphertext modulo q into another ciphertext modulo q using the above transformation process. The resulting ciphertext is then tested using a so-called *quality test*. The key observation is that whilst the ciphertext resulting from this transformation and the input ciphertext are both ciphertexts modulo q encrypting the same plaintext message, they do not necessarily lead to the same drift noise. In particular the drift vector of the input and output ciphertexts from the transform are different, and (by our second important observation mentioned above) can be publicly computed. The purpose of the quality test is to predict a measure on the expected drift noise of mod-switching a ciphertext, knowing the corresponding drift vector.

Let T denote the maximum allowed bound on a certain measure of the drift noise. Two types of quality tests (or a combination thereof) may be used.

- *Probabilistic Quality Test:* Such a quality test may be used when the drift noise depends on the private key. The ciphertext modulo q being tested is fixed, only the private key is unknown. A probabilistic quality test estimates a measure on the drift noise by running over the random choices of the private key for the *fixed* ciphertext being tested and associated drift vector. If the resulting measure is smaller than or equal to T, the test is declared successful. A probabilistic quality test may suffer for some inaccuracies on the measure of the drift noise, but this should occur with a probability that can be set to an arbitrarily small value; in particular, at most p_{err}.

– *Worst-case Quality Test:* Such a quality test may also be used when the drift error depends on the private key. A worst-case quality test estimates a measure on the drift noise by assuming the worst private key (regarding the drift noise) for the *fixed* ciphertext being tested and associated drift vector. Worst-case means that the measure on the drift error cannot be larger for another choice of the private key. In some sense, the test is therefore exact as when it is satisfied, the actual measure on the drift noise is guaranteed to be smaller than or equal to T, independently of the value of the private key.

In all cases, when the test fails, another candidate ciphertext modulo q may be tried via the transformation method above.

5.3 Implementations

Having described the basic philosophy we now examine the two potential quality tests and consider their advantages and disadvantages.

Probabilistic Drift Defense: This is the variant which is probably most practical to be used in deployed FHE schemes. For a *fixed* input LWE-type ciphertext $\boldsymbol{c} = (a_1, \ldots, a_n, b)$ with associated drift vector $(\alpha_1, \ldots, \alpha_n, \beta)$, the corresponding drift noise is $e_{\text{drift}} = \beta - \sum_{j=1}^{n} \alpha_j \cdot s_j$. It is important to note that the ciphertext $\boldsymbol{c}$ is fixed. Over the random choices of the private key $\boldsymbol{s} = (s_1, \ldots, s_n)$, the expectation and variance of the drift noise are given by

$$\mu \coloneqq \mathbb{E}[e_{\text{drift}}] = \beta - \sum_{j=1}^{n} \alpha_j \cdot \mathbb{E}[s_j] \tag{5.1}$$

and

$$\sigma^2 \coloneqq \text{Var}(e_{\text{drift}}) = \sum_{j=1}^{n} \alpha_j^2 \cdot \text{Var}(s_j) \ . \tag{5.2}$$

For example, when $\boldsymbol{s} = (s_1, \ldots, s_n)$ where the s_j's are chosen uniformly at random in $\{0, 1\}$, one has $\mathbb{E}[s_j] = \frac{1}{2}$ and $\text{Var}(s_j) = \frac{1}{4}$ and, consequently, $\mu = \beta - \frac{1}{2}\sum_{j=1}^{n} \alpha_j$ and $\sigma^2 = \frac{1}{4}\sum_{j=1}^{n} \alpha_j^2$. The drift noise in this case is a value in $\mathbb{Z}$. What matters is that it is not too large in absolute value. The relevant measure for the drift noise is therefore the absolute value.

Let T denote the bound on the maximum allowed drift noise (in absolute value). This gives rise to the following *probabilistic* defense given in Figure 5.1, where r is a parameter associated to the desired error probability.

It should be noted that $|\mu| + r \cdot \sigma = \max(|\mu + r \cdot \sigma|, |\mu - r \cdot \sigma|)$. For example, assuming a normal distribution (which is a heuristic assumption that we will revisit in Section 6.3), the actual drift noise (i.e., obtained with the actual private key $\boldsymbol{s}$) corresponding to a drift vector successfully passing this first quality test will lie in the interval $[\mu - r \cdot \sigma, \mu + r \cdot \sigma] \subseteq [-T, T]$ with probability $1 - \text{erfc}(r/\sqrt{2})$. Thus if one wanted an error probability on the output to be at least $1 - 2^{-128}$, then one would select the value r in the algorithm to be greater than 13.11.

Exact Drift Defense: Our second drift defense considers the maximum drift noise a *fixed* input LWE-type $\boldsymbol{c} = (a_1, \ldots, a_n, b)$ with associated drift vector $(\alpha_1, \ldots, \alpha_n, \beta)$ can have. We can assume, without loss of generality, that the key digits s_j are drawn in the integer range $[\![S^-, S^+]\!]$ where $S^- \leq 0$ and $S^+ > 0$. Since the drift noise $e_{\text{drift}} = \beta - \sum_{j=1}^{n} \alpha_j \cdot s_j$ is a signed value, two sub-cases can be distinguished:

Probabilistic Defense

$\mathsf{cnt} \leftarrow 0$;
repeat
  $\boldsymbol{c}' \leftarrow \mathrm{Transform}\big(\boldsymbol{c}, (\boldsymbol{D}_1, \ldots, \boldsymbol{D}_Z), \mathsf{cnt}\big)$;
  Compute the drift vector $(\alpha_1, \ldots, \alpha_n, \beta)$ associated to $\boldsymbol{c}'$;
  $\mu \leftarrow \beta - \sum_{j=1}^{n} \alpha_j \cdot \mathbb{E}[s_j]$;
  $\sigma^2 \leftarrow \sum_{j=1}^{n} \alpha_j^2 \cdot \mathrm{Var}(s_j)$;
until $|\mu| + r \cdot \sigma \leq T$;
return $\boldsymbol{c}'$

Fig. 5.1: Algorithm to transform a ciphertext into one which probabilistically has drift noise bounded by T.

– Maximally negative drift noise: when $s_j = S^+$ for $\alpha_j > 0$ and $s_j = S^-$ for $\alpha_j < 0$, one then has $e_{\mathrm{drift}} = \beta - \sum_{\alpha_j > 0} \alpha_j \cdot S^+ - \sum_{\alpha_j < 0} \alpha_j \cdot S^-$;
– Maximally positive drift noise: when $s_j = S^-$ for $\alpha_j > 0$ and $s_j = S^+$ for $\alpha_j < 0$; one then has $e_{\mathrm{drift}} = \beta - \sum_{\alpha_j > 0} \alpha_j \cdot S^- - \sum_{\alpha_j < 0} \alpha_j \cdot S^+$.

Hence, defining sets $\mathfrak{J}^+ = \big\{j \in [\![1, n]\!] \,|\, \alpha_j > 0\big\}$ and $\mathfrak{J}^- = \big\{j \in [\![1, n]\!] \,|\, \alpha_j < 0\big\}$, the maximal drift noise in absolute value is bounded by

$$M = \max\Big(\big|\beta - S^+ \textstyle\sum_{j \in \mathfrak{J}^+} \alpha_j - S^- \sum_{j \in \mathfrak{J}^-} \alpha_j\big|, \big|\beta - S^- \textstyle\sum_{j \in \mathfrak{J}^+} \alpha_j - S^+ \sum_{j \in \mathfrak{J}^-} \alpha_j\big|\Big) \ .$$

For example, when $\boldsymbol{s} = (s_1, \ldots, s_n)$ where the s_j's are in $\{0, 1\}$ (i.e., for $S^- = 0$ and $S^+ = 1$), the bound M simplifies to

$$M = \max\Big(\big|\beta - \textstyle\sum_{j \in \mathfrak{J}^+} \alpha_j\big|, \big|\beta - \sum_{j \in \mathfrak{J}^-} \alpha_j\big|\Big) \ .$$

Again, letting T denote the bound on the maximum allowed drift noise, this leads to the *exact* defense depicted in Figure 5.2.

Exact Defense

$\mathsf{cnt} \leftarrow 0$;
repeat
  $\boldsymbol{c}' \leftarrow \mathrm{Transform}\big(\boldsymbol{c}, (\boldsymbol{D}_1, \ldots, \boldsymbol{D}_Z), \mathsf{cnt}\big)$;
  $\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$;
  Compute the drift vector $(\alpha_1, \ldots, \alpha_n, \beta)$ associated to $\boldsymbol{c}'$;
  Compute the sets

$$\mathfrak{J}^+ \leftarrow \big\{j \in [\![1, n]\!] \,|\, \alpha_j > 0\big\} \quad \text{and} \quad \mathfrak{J}^- \leftarrow \big\{j \in [\![1, n]\!] \,|\, \alpha_j < 0\big\};$$

  $M \leftarrow \max\Big(\big|\beta - \textstyle\sum_{j \in \mathfrak{J}^+} \alpha_j\big|, \big|\beta - \sum_{j \in \mathfrak{J}^-} \alpha_j\big|\Big)$;
until $M \leq T$;
return $\boldsymbol{c}'$

Fig. 5.2: Algorithm to transform a ciphertext into one which *has* drift noise bounded by T.

This second defense allows one to guarantee that the drift noise of the output ciphertext is always below the maximum bound T (i.e., with probability one), regardless of the actual value of the private key $\boldsymbol{s} = (s_1, \ldots, s_n)$.

5.4 Applications to sIND-CPA^{D} Security of TFHE

Recall from Section 4 that the drift error is the main source of error in TFHE and thus dominates p_{err} as already observed in [15]. For simplicity, for the remainder of this subsection, we therefore assume that the other part of the error e_{in} only has negligible impact on p_{err} and we will ignore it. When considering the sIND-CPA^{D} security of TFHE, we need to take into account the worst-case encryption randomness, meaning that an adversary may choose the vector $\boldsymbol{a}$ in order to maximize the drift error. This leads to significantly larger error bounds and worse parameters compared to the IND-CPA^{D} model.

Assume we are analyzing the public key version of TFHE [24], for which the sIND-CPA^{D} model is more relevant. As noted above, by using the encryption algorithm one may re-randomize the ciphertext without changing the underlying plaintext. Applying Proposition 3.5, we can easily transform an IND-CPA^{D} secure instantiation of public key TFHE into one that is sIND-CPA^{D} secure with the same security level.[4]

Finally, it is noteworthy that the countermeasures introduced in the previous subsections facilitate an analysis of the modulus switching error under adversarially chosen encryption randomness, as done in the next section.

6 Analysis and Experimental Results

In this section, we analyze the proposed drift defenses and derive concrete parameters. Our analysis relies on well-established heuristic assumptions and are backed up by simulations that demonstrate the fit with real measurements.

As proved in Lemma 4.1, the components of the drift vector $(\alpha_1, \ldots, \alpha_n, \beta)$ are considered as independent (identically distributed) samples of a random variable α following the uniform distribution $\mathcal{U}\big(\llbracket -A, A-1 \rrbracket\big)$, where $A \triangleq \frac{q}{4N}$.

We focus on binary secret keys $(s_1, \ldots, s_n)$, where $s_i \sim \mathcal{U}\big(\{0,1\}\big)$, since it turns out that minimizing the variance of s is the best choice in practice. Thus, we plug $\mathbb{E}[s] = \frac{1}{2}$, $\mathrm{Var}(s) = \frac{1}{4}$, and $S^- = 0$, $S^+ = 1$, into the halting conditions used in the probabilistic and exact drift defenses defined in Figures 5.1 and 5.2. These analyses readily adapt to other key distributions.

For both analyses, we will use the following heuristic, motivated by the Central Limit theorem [5, Theorem 27.1]. A quantitative version is given by the Berry–Esseen inequality [4], and experiments show it holds tightly in our cases.

Heuristic 6.1. Let $X_1, \ldots, X_n$ be independent, identically distributed, *bounded*, random variables. For large enough n, it is assumed that

$$\Big(\textstyle\sum_{i=1}^{n} X_i\Big) \sim \mathcal{N}\big(n \cdot \mathbb{E}[X], n \cdot \mathrm{Var}(X)\big) \ .$$

[4] Note, of course, that for this transformation to maintain the security level, the impact of the remaining part of the noise, e_{in}, on p_{err} must be negligible, even in the worst case, and fresh ciphertexts must be perfectly correct. These are mild requirements that can be easily met by an appropriate selection of parameters.

6.1 Probabilistic Test

We analyze here the behavior of the probabilistic defense described in Figure 5.1. For simplicity and as $\mathbb{E}[s] \neq 0$ for binary secrets, we also ignore the role of β which only represents a $\frac{1}{n \cdot \mathbb{E}[s]}$-fraction of the first sum involved in the test.

Therefore, let $Y = \left|\sum_{i=1}^{n} \alpha_i\right|$ and $Z = \sqrt{\sum_{i=1}^{n} \alpha_i^2}$. We want to analyze, for a *fixed* value r to be determined later, the probability density $\varphi_X(t)$ of the random variable $X \coloneqq \mathbb{E}[s] \cdot Y + r\sqrt{\mathrm{Var}(s)} \cdot Z$. This allows setting precisely r so that

$$\Pr\left[\mathbb{E}[s] \cdot Y + r\sqrt{\mathrm{Var}(s)} \cdot Z \leq T\right] = \int_{t=0}^{T} \varphi_X(t)\,\mathrm{d}t \approx p\,, \tag{6.1}$$

where $p \in]0,1]$ is a success probability that ultimately depends on the targeted number of trials for the application. We begin by giving in Lemmas 6.2 and 6.3 the probability densities of Y and Z assuming Heuristic 6.1.

Lemma 6.2. *Assuming Heuristic 6.1, the probability density function φ_Y of the random variable $Y = \left|\sum_{i=1}^{n} \alpha_i\right|$ is given by*

$$\varphi_Y(t) = \mathbb{1}_{t \geq 0} \cdot \frac{1}{\sqrt{2\pi \cdot n\,\mathrm{Var}(\alpha)}} \cdot \left(\exp\left(-\frac{(t - n\mathbb{E}[\alpha])^2}{2 \cdot n\,\mathrm{Var}(\alpha)}\right) + \exp\left(-\frac{(t + n\mathbb{E}[\alpha])^2}{2 \cdot n\,\mathrm{Var}(\alpha)}\right)\right)\,.$$

Proof. By definition, $\Phi_Y(t) \coloneqq \Pr\left[Y \leq t\right] = \Pr\left[\sum_{i=1}^{n} \alpha_i \in [-t, t]\right]$, thus Heuristic 6.1 implies $\Phi_Y(t) = \Phi\left(\frac{t-\mu}{\sigma}\right) - \Phi\left(\frac{-t-\mu}{\sigma}\right)$, where $\mu = n\mathbb{E}[\alpha]$ and $\sigma^2 = n\,\mathrm{Var}(\alpha)$. Deriving this expression for t yields the result. □

Lemma 6.3. *Assuming Heuristic 6.1, the probability density function φ_Z of the random variable $Z = \sqrt{\sum_{i=1}^{n} \alpha_i^2}$ is given by*

$$\varphi_Z(t) = \mathbb{1}_{t \geq 0} \cdot \frac{1}{\sqrt{2\pi \cdot n\,\mathrm{Var}(\alpha^2)}} \cdot 2t \cdot \exp\left(-\frac{(t^2 - n\mathbb{E}[\alpha^2])^2}{2n\,\mathrm{Var}(\alpha^2)}\right)\,.$$

Proof. Likewise, $\Phi_Z(t) \coloneqq \Pr\left[Z \leq t\right] = \Pr\left[0 \leq \sum_{i=1}^{n} \alpha_i^2 \leq t^2\right]$, which by Heuristic 6.1 is $\Phi\left(\frac{t^2 - n\mathbb{E}[\alpha^2]}{\sqrt{n\,\mathrm{Var}(\alpha^2)}}\right) - \Phi\left(\frac{-n\mathbb{E}[\alpha^2]}{\sqrt{n\,\mathrm{Var}(\alpha^2)}}\right)$, and deriving for t gives the result. □

In order to obtain the probability distribution function of X, we make another heuristic approximation, which is not formally true for non-normal distributions [30], but yields accurate results in our practical setting.

Heuristic 6.4. Let U be a random variable following a uniform distribution. For a large enough (fixed) number of samples, it is assumed that the sampled mean and variance of U behave as independent random variables.

Proposition 6.5. *Assuming Heuristics 6.1 and 6.4, the probability density function φ_X of the random variable $X = \mathbb{E}[s] \cdot Y + r\sqrt{\mathrm{Var}(s)} \cdot Z$ is given by*

$$\varphi_X(t) = \frac{1}{\mathbb{E}[s] \cdot r\sqrt{\mathrm{Var}(s)}} \cdot \int_{z=0}^{t} \varphi_Y\left(\frac{t-z}{\mathbb{E}[s]}\right) \cdot \varphi_Z\left(\frac{z}{r\sqrt{\mathrm{Var}(s)}}\right)\mathrm{d}z\,.$$

Proof. Heuristic 6.4 allows considering $\mathbb{E}[s]Y$ and $r\sqrt{\mathrm{Var}(s)}Z$ as independent variables, thus the probability distribution function of their sum is the convolution of $\varphi_{\mathbb{E}[s]\cdot Y}$ and $\varphi_{r\sqrt{\mathrm{Var}(s)}\cdot Z}$. Both can be derived from Lemmas 6.2 and 6.3, using that for any $u \neq 0$ and random variable R, $\varphi_{u \cdot R}(t) = \frac{1}{u}\varphi_R(\frac{t}{u})$. Finally, as these densities are 0 for negative values, the convolution ranges over $]0, t[$. □

Obtaining a closed form for φ_X seems rather involved. Nevertheless, it is reasonably easy to evaluate numerically, which is sufficient in practice to check the accuracy of Heuristics 6.1 and 6.4 and to set r according to Equation (6.1).

Approximation by a Skew Normal Distribution: We show that the probability density function given in Proposition 6.5 is close to the probability density function of a skew normal distribution [3].

For simplicity,[5] we consider that α is centered around 0, i.e., $\mathbb{E}[\alpha] = 0$; this is justified by the fact that $\mathbb{E}[\alpha]$ is in practice very negligible compared to other quantities involved. We further assume the following natural[6] heuristic.

Heuristic 6.6. Let N be a random variable following a normal distribution of variance $\mathrm{Var}(N)$ centered at $\mathbb{E}[N]$. Provided that $\mathbb{E}[N] > 0$ and that $\frac{\mathbb{E}[N]}{\sqrt{\mathrm{Var}(N)}}$ is large enough (e.g., more than a few tens), it is assumed that

$$\sqrt{N} \sim \mathcal{N}\Big(\sqrt{\mathbb{E}[N] - \tfrac{\mathrm{Var}(N)}{4\mathbb{E}[N]}}, \tfrac{\mathrm{Var}(N)}{4\mathbb{E}[N]}\Big) \ .$$

Justification (Heuristic 6.6). Following the reasoning in the proof of Lemma 6.3,

$$\Pr[0 \le \sqrt{N} \le t] = \int_{u=0}^{t} \tfrac{1}{\sqrt{2\pi\,\mathrm{Var}(N)}} \cdot 2u \cdot \exp -\tfrac{1}{2}\Big(\tfrac{u^2-\mathbb{E}[N]}{\sqrt{\mathrm{Var}(N)}}\Big)^2 \mathrm{d}u \ ,$$

which after the change of variable $v = \frac{u^2-\mathbb{E}[N]}{\sqrt{4\mathbb{E}[N]}} + \sqrt{\mathbb{E}[N] - \frac{\mathrm{Var}(N)}{4\mathbb{E}[N]}}$ gives

$$= \int_{v=v_1}^{v_2} \sqrt{\tfrac{4\mathbb{E}[N]}{2\pi\,\mathrm{Var}(N)}} \exp -\tfrac{1}{2}\Big(\tfrac{v-\sqrt{\mathbb{E}[N]-\mathrm{Var}(N)/(4\mathbb{E}[N])}}{\sqrt{\mathrm{Var}(N)/(4\mathbb{E}[N])}}\Big)^2 \mathrm{d}v \ .$$

This is the intended normal distribution, although integrated between the new bounds $v_1 \coloneqq \sqrt{\mathbb{E}[N]}\Big(-\frac{1}{2} + \sqrt{1 - \frac{\mathrm{Var}(N)}{4\mathbb{E}[N]^2}}\Big)$ and $v_2 \coloneqq v_1 + \frac{t^2}{2\sqrt{\mathbb{E}[N]}}$. As the difference between v_1 and the target expectation is exactly $-\frac{\sqrt{\mathbb{E}[N]}}{2}$, the lower $\int_{-\infty}^{v_1}$ is equal to $\frac{1}{2} - \frac{1}{2}\,\mathrm{erf}\Big(\frac{\mathbb{E}[N]/\sqrt{\mathrm{Var}(N)}}{\sqrt{2}}\Big)$, which is negligible by hypothesis. Furthermore, it is easy to verify that

$$t - v_2 \le \sqrt{\mathbb{E}[N]}\Big(1 - \sqrt{1 - \tfrac{\mathrm{Var}(N)}{4\mathbb{E}[N]^2}}\Big) \approx \sqrt{\mathbb{E}[N]} \cdot \tfrac{\mathrm{Var}(N)}{8\mathbb{E}[N]^2} \ ,$$

thus $\int_{v_2}^{t} \le (t - v_2) \cdot \sqrt{\frac{4\mathbb{E}[N]}{2\pi\,\mathrm{Var}(N)}} \approx \frac{1}{4\sqrt{2\pi}} \cdot \frac{\sqrt{\mathrm{Var}(N)}}{\mathbb{E}[N]}$, less than 0.5% in practice. □

Combined with Heuristic 6.1, Heuristic 6.6 assimilates the distribution of Z given by Lemma 6.3 to $\mathcal{N}\Big(\sqrt{n\mathbb{E}[\alpha^2] - \frac{\mathrm{Var}(\alpha^2)}{4\mathbb{E}[\alpha^2]}}, \frac{\mathrm{Var}(\alpha^2)}{4\mathbb{E}[\alpha^2]}\Big)$. We stress that, when the bound A on $|\alpha|$ is not singularly small, we have $\frac{\mathbb{E}[\alpha^2]}{\sqrt{\mathrm{Var}(\alpha^2)}} \approx \frac{\sqrt{5}}{2} \cdot \sqrt{n}$, which is more than enough in practice for applying the heuristic.

Proposition 6.7. *Assuming Heuristics 6.1, 6.4 and 6.6 with $\mathbb{E}[\alpha] = 0$, the probability density function of $X = \mathbb{E}[s] \cdot Y + r\sqrt{\mathrm{Var}(s)} \cdot Z$ is approximated by*

$$\varphi_{\mathcal{S}}(t) = \frac{2}{\omega} \cdot \varphi\Big(\frac{t-\xi}{\omega}\Big) \cdot \Phi\Big(\varrho\Big(\frac{t-\xi}{\omega}\Big)\Big) \ ,$$

[5] This primarily allows one to avoid splitting φ_Y from Lemma 6.2 in two equivalent parts and avoids dragging around translation terms ($t \pm n\mathbb{E}[\alpha]$) in the computations.

[6] Recall that by definition of the variance, $\mathrm{Var}(\sqrt{N}) = \mathbb{E}[N] - \mathbb{E}[\sqrt{N}]^2$.

which is the probability density function of a random variable $\mathcal{S}$ *following a* skew normal distribution *of localisation* ξ*, scale* ω *and shape* ϱ*, where*

$$\varrho = \frac{\mathbb{E}[s]}{r\sqrt{\operatorname{Var}(s)}} \cdot \sqrt{\frac{n\operatorname{Var}(\alpha)}{\operatorname{Var}(\alpha^2)/(4\mathbb{E}[\alpha^2])}}\,, \qquad \omega = \mathbb{E}[s]\sqrt{n\operatorname{Var}(\alpha)} \cdot \sqrt{1+\frac{1}{\varrho^2}}\,,$$
$$\xi = r\sqrt{\operatorname{Var}(s)} \cdot \sqrt{n\mathbb{E}[\alpha^2] - \frac{\operatorname{Var}(\alpha^2)}{4\mathbb{E}[\alpha^2]}}\,.$$

Proof. Assuming $\mathbb{E}[\alpha] = 0$ and Heuristic 6.6, the convolution probability density function given by Proposition 6.5 under Heuristics 6.1 and 6.4 simplifies to

$$\varphi_{\mathcal{S}}(t) = \int_{y=-\infty}^{t} \frac{2/\mathbb{E}[s]}{\sqrt{2\pi\cdot\mathbb{V}_Y}} \exp\left(-\frac{1}{2}\left(\frac{t-y}{\mathbb{E}[s]\cdot\sqrt{\mathbb{V}_Y}}\right)^2\right) \cdot \frac{1/\mathbb{r}_s}{\sqrt{2\pi\cdot\mathbb{V}_Z}} \cdot \exp\left(-\frac{1}{2}\left(\frac{y-\mathbb{r}_s\mathbb{E}_Z}{\mathbb{r}_s\cdot\sqrt{\mathbb{V}_Z}}\right)^2\right) \mathrm{d}y\,,$$

where $\mathbb{V}_Y \coloneqq n\operatorname{Var}(\alpha)$, $\mathbb{E}_Z \coloneqq \sqrt{n\mathbb{E}[\alpha^2] - \frac{\operatorname{Var}(\alpha^2)}{4\mathbb{E}[\alpha^2]}}$ and $\mathbb{V}_Z \coloneqq \frac{\operatorname{Var}(\alpha^2)}{4\mathbb{E}[\alpha^2]}$, according to Lemma 6.2 and Heuristic 6.6, and where $\mathbb{r}_s \coloneqq r\sqrt{\operatorname{Var}(s)}$ for concision. In particular, with these notations we have $\xi = \mathbb{r}_s \cdot \mathbb{E}_Z$.

Now, notice that by definition of ω and ϱ, $\mathbb{E}[s]\sqrt{\mathbb{V}_Y} = \omega\sqrt{1-\frac{1}{1+\varrho^2}}$ and likewise, $\mathbb{r}_s\sqrt{\mathbb{V}_Z} = \omega\sqrt{\frac{1}{1+\varrho^2}}$ using $\left(1+\frac{1}{\varrho^2}\right)\frac{1}{1+\varrho^2} = \frac{1}{\varrho^2}$. Grouping together the terms inside the exponentials and rearranging them yield, after a few calculations,

$$-\frac{1}{2}\left[\frac{\big((t-\xi)-(y-\xi)\big)^2}{\omega^2\cdot\left(1-\frac{1}{1+\varrho^2}\right)} + \frac{(y-\xi)^2}{\omega^2\cdot\frac{1}{1+\varrho^2}}\right] = -\frac{1}{2}\left[\left(\frac{t-\xi}{\omega}\right)^2 + \left(\frac{(1+\varrho^2)(y-t)+\varrho^2(t-\xi)}{\omega\varrho}\right)^2\right]\,.$$

The first summand above can be extracted from the integral, and applying the change of variable $u = \frac{(1+\varrho^2)y-\varrho^2\xi-t}{\omega\varrho}$ to the second summand finally gives

$$\varphi_{\mathcal{S}}(t) = \frac{2}{\omega\sqrt{2\pi}} \cdot \exp\left(-\frac{1}{2}\left(\frac{t-\xi}{\omega}\right)^2\right) \cdot \int_{u=-\infty}^{\varrho\left(\frac{t-\xi}{\omega}\right)} \frac{1}{\sqrt{2\pi}} \exp -\frac{u^2}{2}\,\mathrm{d}u\,,$$

using for the constant terms that $\frac{2/\mathbb{E}[s]}{\sqrt{\mathbb{V}_Y}} = \frac{2}{\omega}\cdot\sqrt{1+\frac{1}{\varrho^2}}$ and $\frac{1/\mathbb{r}_s}{\sqrt{\mathbb{V}_Z}} = \frac{\sqrt{1+\varrho^2}}{\omega}$, which combine perfectly to build $\mathrm{d}u = \frac{1+\varrho^2}{\omega\varrho}\,\mathrm{d}y$. □

The skew normal distribution given by Proposition 6.7 has been well-studied. In particular, the antiderivative of $\varphi_{\mathcal{S}}(t)$ is given by $\Phi\left(\frac{t-\xi}{\omega}\right) - 2\mathcal{T}\left(\frac{t-\xi}{\omega}, \varrho\right)$, where Owen's $\mathcal{T}$ function [32] can be efficiently evaluated (see e.g., [25]). Thus, for any r, the probability $p(r)$ of passing the quality test is immediately estimated by

$$p(r) \approx \int_0^T \varphi_{\mathcal{S}}(t)\,\mathrm{d}t = \Phi\left(\frac{T-\xi}{\omega}\right) - 2\mathcal{T}\left(\frac{T-\xi}{\omega}, \varrho\right)\,. \tag{6.2}$$

Given a targetted expected number of trials $1/p$, this allows obtaining quick estimates of a value r such that $p(r) \approx p$ using binary search.

6.2 Worst-Case Test

In order to analyze the exact defense defined in Figure 5.2, the components of the drift vector are partitioned between the positive and negative ones. Thus, each of the $|\alpha_i|$ is considered as an independent sample of a uniform variable $|\alpha|$ following the uniform distribution $\mathcal{U}(\llbracket 0, A \rrbracket)$, with $\mathbb{E}\big[|\alpha|\big] \approx \frac{q}{8N}$ and $\mathrm{Var}\big(|\alpha|\big) \approx \frac{q^2}{192N^2}$.

For similar reasons as in Section 6.1, we also ignore β. Our experiments show that this still yields a fairly close approximation. We start by observing that for a given bound T we have

$$\begin{aligned}\Pr[M \leq T] &= \Pr\Big[\big(\textstyle\sum_{i\in\mathfrak{J}^+}|\alpha_i| \leq T\big) \wedge \big(\sum_{i\in\mathfrak{J}^-}|\alpha_i| \leq T\big)\Big] \\ &= \sum_{j=0}^{n} \Pr\big[\sharp\mathfrak{J}^+ = j\big] \\ &\qquad \cdot \Pr\Big[\big(\textstyle\sum_{i\in\mathfrak{J}^+}|\alpha_i| \leq T\big) \wedge \big(\sum_{i\in\mathfrak{J}^-}|\alpha_i| \leq T\big) \;\Big|\; \sharp\mathfrak{J}^+ = j\Big] \;.\end{aligned}$$

Conditioned on a fixed size of $\mathfrak{J}^+$ (thus of $\mathfrak{J}^-$), the two sums are independent:

$$\begin{aligned}\Pr[M \leq T] = \sum_{j=0}^{n} \Pr\big[\sharp\mathfrak{J}^+ = j\big] &\cdot \Pr\big[\textstyle\sum_{i\in\mathfrak{J}^+}|\alpha_i| \leq T \;\big|\; \sharp\mathfrak{J}^+ = j\big] \\ &\cdot \Pr\big[\textstyle\sum_{i\in\mathfrak{J}^-}|\alpha_i| \leq T \;\big|\; \sharp\mathfrak{J}^- = n - j\big] \;.\end{aligned}$$

As long as both sets $\mathfrak{J}^+$ and $\mathfrak{J}^-$ are large enough, we can apply Heuristic 6.1 in order to approximate the distribution of $\sum_{i\in\mathfrak{J}^+}|\alpha_i|$ (resp. $\sum_{i\in\mathfrak{J}^-}|\alpha_i|$) by a normal distribution of variance $\sharp\mathfrak{J}^{\pm} \cdot \mathrm{Var}(|\alpha|)$ centered at $\sharp\mathfrak{J}^{\pm} \cdot \mathbb{E}[|\alpha|]$.

This leaves the case when either $\sharp\mathfrak{J}^+$ or $\sharp\mathfrak{J}^-$ is small. Luckily, the probability of this event can be easily bounded using Chernoff bound. For this, notice that $\sharp\mathfrak{J}^+$ as a random variable follows the binomial distribution $\mathcal{B}\big(n, \frac{1}{2}\big)$. The Chernoff bound applied to $\sharp\mathfrak{J}^+$ yields, for $0 < \delta < 1$,

$$\Pr\Big[\big|\sharp\mathfrak{J}^+ - n/2\big| > \delta n/2\Big] \leq 2e^{-\frac{\delta^2 n}{6}} \;.$$

Hence, for any (small) $\epsilon > 0$, we can set $\delta = \sqrt{6\ln(2/\epsilon)/n}$, which in turn ensures that $\Pr\big[\big|\sharp\mathfrak{J}^+ - n/2\big| \geq \delta n/2\big] \leq \epsilon$. Putting things together with Equation (2.1), we thus obtain $p \leq \Pr[M \leq T] \leq p + \epsilon$, where

$$p = \frac{1}{2^{n+2}} \sum_{j=(1-\delta)\frac{n}{2}}^{(1+\delta)\frac{n}{2}} \binom{n}{j} \cdot \mathrm{erfc}\left(\frac{\frac{jq}{8N} - T}{\sqrt{\frac{j}{96}\frac{q}{N}}}\right) \cdot \mathrm{erfc}\left(\frac{\frac{(n-j)q}{8N} - T}{\sqrt{\frac{n-j}{96}\frac{q}{N}}}\right) \;. \tag{6.3}$$

Whilst this is not a very elegant formula, it can be easily numerically computed for concrete parameters and extensive experiments suggest it is accurate.

Numerical Experiments: In order to verify the bound given by Equation (6.3), we considered the following setup: we applied the technique to a random ciphertext until the result was smaller than T, and averaged over 50 experiments the measured number of required re-randomizations.

For sample parameters $q = 2^{64}$, $n = 841$, $N = 2048$ and $T = 2^{58.65}$, Equation (6.3) yields $p \approx 2^{-9.2}$. Over 50 experiments the measured average number of re-randomizations was about $2^{9.4}$, which indicates that our estimation for p is fairly accurate. In comparison, in order to bound the error with probability 1 *without* the drift defense, we need to assume worst-case bounds for all α_i and β combined with a worst-case key, which yields $T = \frac{(n+1)q}{4N} \approx 2^{60.7}$. This is more than four times larger and therefore leads to less performant parameter choices.

6.3 Validating the Heuristics and Concrete Parameters

In this subsection, we experimentally validate the accuracy of the heuristics used in the analysis of the probabilistic drift defense, and study the practical impact of this new technique on existing sets of parameters *for binary secrets*.

We consider sets of parameters taken from the `TFHE-rs` library [38] and designed for boolean and 4-bit payloads. Typically, these parameters are selected so as to ensure that a ciphertext resulting from a programmable bootstrapping correctly decrypts to the expected plaintext except with probability at most p_{err}. We stress that the `TFHE-rs` library has been designed to obtain IND-CPA security, *not* with IND-CPA^{D} security in mind, thus error probabilities are set to be less than 2^{-128}. In a recent update,[7] the value of p_{err} in the `TFHE-rs` library has been lowered to 2^{-64}. For completeness and to ease the comparison with other papers as, e.g., [15], we also include the corresponding *outdated* parameters[8] that were designed for $p_{\text{err}} < 2^{-40}$. This is summarized in Table 6.1.

Table 6.1: Considered sets of parameters from `TFHE-rs` library [38] for boolean and 4-bit payloads. Parameters marked with † ($p_{\text{err}} \approx 2^{-40}$) are *outdated*.

Set	Name in `TFHE-rs`	q	n	N	k	p_{err}	r_0
$\mathfrak{Z}_{\text{b}}^{40\dagger}$	`DEFAULT_PARAMETERS`	2^{32}	722	512	2	2^{-40}	7.15
$\mathfrak{Z}_{\text{4b}}^{40\dagger}$	`PARAM_MESSAGE_2_CARRY_2_KS_PBS_GAUSSIAN_2M40`	2^{64}	761	2048	1	2^{-40}	7.15
$\mathfrak{Z}_{\text{b}}^{64}$	`DEFAULT_PARAMETERS_KS_PBS`	2^{64}	739	512	3	2^{-64}	9.16
$\mathfrak{Z}_{\text{4b}}^{64}$	`PARAM_MESSAGE_2_CARRY_2_KS_PBS_GAUSSIAN_2M64`	2^{64}	834	2048	1	2^{-64}	9.16

In particular, the value of p_{err} is set so as to ensure that the absolute value of the drift error is below a certain threshold $T = r_0\sqrt{\text{Var}(e_{\text{drift}})}$, where $\text{Var}(e_{\text{drift}})$ is given by Equation (4.3) and r_0 verifies $p_{\text{err}} = \text{erfc}\left(\frac{r_0}{\sqrt{2}}\right)$. In this section we aim to show that the parameters chosen in `TFHE-rs` to achieve IND-CPA security using the traditional modulus switch can also achieve *strong*-IND-CPA^{D} security with our new modulus switching operations, incurring only a negligible overhead.

Validating the Post-Test Distribution Heuristic: In order to convert the threshold parameters in the probabilistic defense into a decryption failure probability after a bootstrapping as done in Section 5, we need to assume that, for a fixed drift vector passing the quality test, the distribution of the drift error *under random binary secret keys* is a normal distribution $\mathcal{N}(\mu, \sigma)$, where μ and σ are given by Equations (5.1) and (5.2) respectively.

To justify this heuristic, we generated several drift vectors based on parameter set $\mathfrak{Z}_{\text{4b}}^{64}$ passing the quality test with $r = 13.4$ and $T = 2^{57.76}$. For each of them, we sampled 2^{16} binary secret keys and plotted the distribution of the drift error against the expected normal distribution. A few examples are shown in Figure 6.2 and additional graphs can be found in Appendix B, which convincingly demonstrate the plausibility of the heuristic assumption.

Validating Statistical Heuristics: In order to validate experimentally the analyses of Section 6.1, and in particular the relevance of Heuristics 6.1, 6.4 and 6.6, we sampled 10^6 random drift vectors (including β) for each set of parameters.

As shown in Figure 6.3 for, e.g., $r = 13.11$, the measured distribution of X matches perfectly the curve of φ_X computed from Proposition 6.5. We also feature its approximation

[7] Git commit `400ce27beb5bea8fdc68826ad437099ec62680d0`, September 25th, 2024.
[8] See, e.g., Git commit `d1fe49fa2fae36d39ba9f779b7b71e785b66c3b2`.

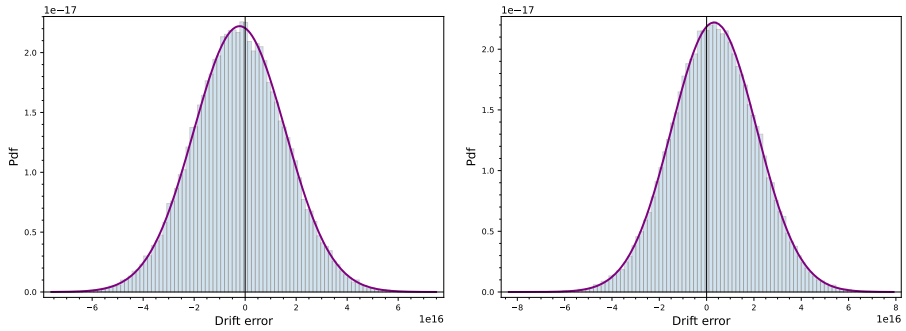


Fig. 6.2: Measured vs. expected drift distribution for several drift vectors passing the quality test for parameter set $\mathfrak{Z}_{4\mathrm{b}}^{64}$ ($T = 2^{57.76}$) and $r = 13.4$.

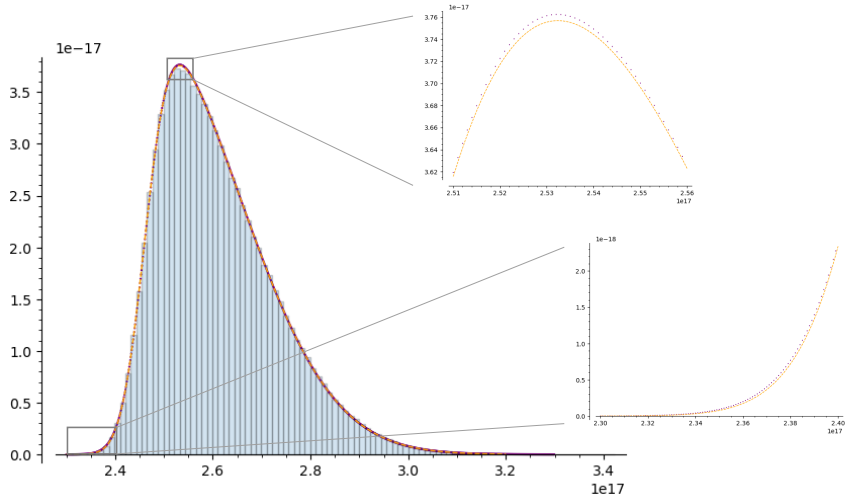


Fig. 6.3: Probability density function of X for parameter set $\mathfrak{Z}_{4\mathrm{b}}^{64}$ and $r = 13.11$: *i.* measured from 10^6 random drift vectors (light blue histogram), *ii.* computed from Pr. 6.5 on 10^3 evaluation points (purple line), and *iii.* approximated by the skew normal distribution from Pr. 6.7 (dashed orange line).

by the skew normal distribution defined by Proposition 6.7; concretely, both curves are superposed.

Furthermore, we compared the measured value of $\mathbb{E}[Z]$ with the estimation given by Heuristic 6.6. This appears in Table 6.4; in practice, the estimation is so accurate that we only report one value for $\mathbb{E}[Z]$.

Concrete Security for a Given Number of Trials: Finally, setting a reasonable target probability of passing the quality test, namely $p \in \left\{\frac{1}{50}, \frac{1}{100}, \frac{1}{1000}\right\}$, we derived the corresponding value of r_{new} using a binary search based either on Proposition 6.5, or on the skew normal distribution from Proposition 6.7. We further validated experimentally that the average number of trials for the obtained r_{new} is indeed close to $1/p$. The resulting probability of failure p_{new}, for the given set of parameters *patched with the new modulus switching technique* using the found r_{new}, is obtained from $p_{\mathrm{new}} = \mathrm{erfc}\left(\frac{r_{\mathrm{new}}}{\sqrt{2}}\right)$.

All these results are detailed in Table 6.4. As a conclusion, the analysis of Section 6.1 is very accurate experimentally. Numerical values show that the new probabilistic drift defense roughly doubles the logarithm of the failure probability of a parameter set using

Table 6.4: Impact of the probabilistic drift defense on `TFHE-rs` parameters, depending on the expected number of trials (p^{-1}) for the quality test with threshold T.

Set	q	n	N	T	$\mathbb{E}[Z]$	p^{-1}	r_{new}	♯ trials	$\log_2 p_{\text{new}}^{-1}$
$\mathfrak{Z}_{\text{b}}^{40\dagger}$	2^{64}	722	512	$2^{27.30}$	$2^{24.96}$	50	10.244	49.0	79.39
						100	10.313	98.3	80.42
						1000	10.495	1008.8	83.18
$\mathfrak{Z}_{\text{4b}}^{40\dagger}$	2^{64}	761	2048	$2^{57.33}$	$2^{54.99}$	50	10.238	48.8	79.30
						100	10.304	99.5	80.30
						1000	10.482	989.0	82.99
$\mathfrak{Z}_{\text{b}}^{64}$	2^{64}	739	512	$2^{59.67}$	$2^{56.97}$	50	13.152	49.1	128.83
						100	13.235	99.3	130.41
						1000	13.459	1014.2	134.75
$\mathfrak{Z}_{\text{4b}}^{64}$	2^{64}	834	2048	$2^{57.76}$	$2^{55.06}$	50	13.132	49.5	128.44
						100	13.210	100.1	129.94
						1000	13.422	1063.7	134.02

only 50 re-randomizations. This is completely unnoticeable, as a modulus switch typically takes $1/150\,000^{\text{th}}$ of the time of the blind rotation.

Acknowledgements: The authors would like to thank Benoît Libert for useful discussions and feedback on the paper.

References

1. Alexandru, A., Badawi, A.A., Micciancio, D., Polyakov, Y.: Application-aware approximate homomorphic encryption: Configuring FHE for practical use. Cryptology ePrint Archive, Report 2024/203 (2024), https://eprint.iacr.org/2024/203
2. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 297–314. Springer, Berlin, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_17
3. Azzalini, A.: A class of distributions which includes the normal ones. Scandinavian Journal of Statistics **12**(5), 171–178 (1985), http://www.jstor.org/stable/4615982
4. Berry, A.C.: The accuracy of the Gaussian approximation to the sum of independent variates. Transactions of the American Mathematical Society **49**(1), 122–136 (1941). https://doi.org/10.1090/S0002-9947-1941-0003498-3
5. Billingsley, P.: Probability and Measure. Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, 3rd edn. (1995)
6. Bonnoron, G., Ducas, L., Fillinger, M.: Large FHE gates from tensored homomorphic accumulator. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 18. LNCS, vol. 10831, pp. 217–251. Springer, Cham (May 2018). https://doi.org/10.1007/978-3-319-89339-6_13
7. Bonte, C., Iliashenko, I., Park, J., Pereira, H.V.L., Smart, N.P.: FINAL: Faster FHE instantiated with NTRU and LWE. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part II. LNCS, vol. 13792, pp. 188–215. Springer, Cham (Dec 2022). https://doi.org/10.1007/978-3-031-22966-4_7
8. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Berlin, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_50
9. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS 2012. pp. 309–325. ACM (Jan 2012). https://doi.org/10.1145/2090236.2090262
10. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky, R. (ed.) 52nd FOCS. pp. 97–106. IEEE Computer Society Press (Oct 2011). https://doi.org/10.1109/FOCS.2011.12
11. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Berlin, Heidelberg (Aug 2011). https://doi.org/10.1007/978-3-642-22792-9_29

12. Canard, S., Fontaine, C., Phan, D.H., Pointcheval, D., Renard, M., Sirdey, R.: Relations among new CCA security notions for approximate FHE. Cryptology ePrint Archive, Report 2024/812 (2024), https://eprint.iacr.org/2024/812
13. Checri, M., Sirdey, R., Boudguiga, A., Bultel, J.P.: On the practical CPA^D security of "exact" and threshold FHE schemes and libraries. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part III. LNCS, vol. 14922, pp. 3–33. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68382-4_1
14. Chen, Z., Wang, J., Chen, L., Song, X.: A Regev-type fully homomorphic encryption scheme using modulus switching. The Scientific Journal **14** (2014). https://doi.org/10.1155/2014/983862, article ID 983862
15. Cheon, J.H., Choe, H., Passelègue, A., Stehlé, D., Suvanto, E.: Attacks against the INDCPA-D security of exact FHE schemes. In: Lie, D., Kirda, E. (eds.) ACM CCS 2024. pp. 2505–2519. ACM Press (Nov 2024). https://doi.org/10.1145/3658644.3690341
16. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 409–437. Springer, Cham (Dec 2017). https://doi.org/10.1007/978-3-319-70694-8_15
17. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (Jan 2020). https://doi.org/10.1007/s00145-019-09319-x
18. Chillotti, I., Joye, M., Paillier, P.: Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In: Dolev, S., et al. (eds.) Cyber Security Cryptography and Machine Learning (CSCML 2021). LNCS, vol. 12716, pp. 1–19. Springer (2021). https://doi.org/10.1007/978-3-030-78086-9_1
19. Ducas, L., Micciancio, D.: FHEW: Bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 617–640. Springer, Berlin, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46800-5_24
20. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), https://eprint.iacr.org/2012/144
21. Gama, N., Izabachène, M., Nguyen, P.Q., Xie, X.: Structural lattice reduction: Generalized worst-case to average-case reductions and homomorphic cryptosystems. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 528–558. Springer, Berlin, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_19
22. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009), https://crypto.stanford.edu/craig
23. Joye, M.: SoK: Fully homomorphic encryption over the [discretized] torus. IACR TCHES **2022**(4), 661–692 (2022). https://doi.org/10.46586/tches.v2022.i4.661-692
24. Joye, M.: TFHE public-key encryption revisited. In: Oswald, E. (ed.) CT-RSA 2024. LNCS, vol. 14643, pp. 277–291. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58868-6_11
25. Komelj, J.: The bivariate normal integral via Owen's T function as a modified Euler's arctangent series. American Journal of Computational Mathematics **13**(4), 476–504 (2023). https://doi.org/10.4236/ajcm.2023.134026
26. Lee, Y., Micciancio, D., Kim, A., Choi, R., Deryabin, M., Eom, J., Yoo, D.: Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 227–256. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30620-4_8
27. Li, B., Micciancio, D.: On the security of homomorphic encryption on approximate numbers. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 648–677. Springer, Cham (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_23
28. Li, B., Micciancio, D., Schultz, M., Sorrell, J.: Securing approximate homomorphic encryption using differential privacy. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part I. LNCS, vol. 13507, pp. 560–589. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15802-5_20
29. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Karloff, H.J., Pitassi, T. (eds.) 44th ACM STOC. pp. 1219–1234. ACM Press (May 2012). https://doi.org/10.1145/2213977.2214086
30. Lukacs, E.: A characterization of the normal distribution. The Annals of Mathematical Statistics **13**(1), 91–93 (1942). https://doi.org/10.1214/aoms/1177731647
31. Manulis, M., Nguyen, J.: Fully homomorphic encryption beyond IND-CCA1 security: Integrity through verifiability. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part II. LNCS, vol. 14652, pp. 63–93. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58723-8_3
32. Owen, D.B.: Tables for computing bivariate normal probabilities. The Annals of Mathematical Statistics **27**(4), 1075–1090 (1956), http://www.jstor.org/stable/2237196
33. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM **56**(6), 34:1–34:40 (2009). https://doi.org/10.1145/1568318.1568324

34. Rothblum, R.: Homomorphic encryption: From private-key to public-key. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 219–234. Springer, Berlin, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19571-6_14
35. Shoup, V.: Why chosen ciphertext security matters. Research Report RZ 3076, IBM (Nov 1998), https://shoup.net/papers/expo.pdf
36. Smart, N.P.: Practical and efficient FHE-based MPC. In: Quaglia, E.A. (ed.) 19th IMA International Conference on Cryptography and Coding. LNCS, vol. 14421, pp. 263–283. Springer, Cham (Dec 2023). https://doi.org/10.1007/978-3-031-47818-5_14
37. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Berlin, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_2
38. Zama: TFHE-rs: A pure Rust implementation of the TFHE scheme for boolean and integer arithmetics over encrypted data (2022), https://github.com/zama-ai/tfhe-rs

A Attacks Against Failing Circuits

In this appendix, we review attacks presented in [15] and specifically apply them in the context of TFHE. The attacks target failing circuits. For illustration purposes, as in [15], we consider boolean circuits with an AND gate.

A.1 A Generic Distinguishing Attack

Suppose a boolean circuit that takes $k+1$ boolean variables $(X, X_1, \ldots, X_k)$ on input and outputs the boolean variable $Z = X \wedge Y$ where $Y \leftarrow \mathcal{C}^*(X_1, \ldots, X_k)$ for some boolean circuit $\mathcal{C}^*$; see Figure A.1. Boolean variable X is private and is unknown to the attacker. The goal for the attacker is to tell with probability non-negligibly larger than $1/2$ if $X = 0$ or if $X = 1$. The attacker can freely choose the values of $(X_1, \ldots, X_k)$ and the sub-circuit $\mathcal{C}^*$.

Consider the specific IND-CPA^{D} security game between a challenger and an attacker for the functions considered in Figure A.1 given in Figure A.2. The challenger chooses a secret bit b. The attacker wins if bit b is correctly guessed at the end of the game. Later we will show that the probability that the attacker correctly guesses b (i.e., $\Pr[b' = b]$) in the above game is *strictly* larger than $\frac{1}{2}$. Recall that a probability of $\frac{1}{2}$ corresponds to a random guess.

A.2 A Key Recovery Attack

The knowledge of failing ciphertexts can be turned into a key recovery attack *once a defective programmable bootstrapping is identified*. As an illustration, suppose we have a simple circuit consisting of a single AND gate.
The AND of two boolean values X and Y can be obtained by returning the most significant bit of $X + Y \pmod 4$. Over encrypted data, evaluating the msb is carried out through a programmable bootstrapping.

We use the notations of Section 4 where recall we defined the drift error as $e_{\text{drift}} = \beta - \sum_{i=1}^{n} \alpha_i \, s_i$. Suppose that a ciphertext incorrectly decrypts. This occurs when the drift error present in the ciphertext is large (i.e., above a certain threshold). Using vector notation, defining the drift vector as $\boldsymbol{\delta} \coloneqq (\alpha_1, \ldots, \alpha_n, \beta)$, the drift error can be rewritten as

$$e_{\text{drift}} = \langle (-\boldsymbol{s}, 1), \boldsymbol{\delta} \rangle \ .$$

A larger drift error means that the secret vector $(-\boldsymbol{s}, 1)$ is more 'aligned' with the drift vector $\boldsymbol{\delta}$. Furthermore, when $s_i = 0$, the corresponding α_i can take any value as it has then no impact on the drift error. These two observations lead to the attack in Figure A.4.

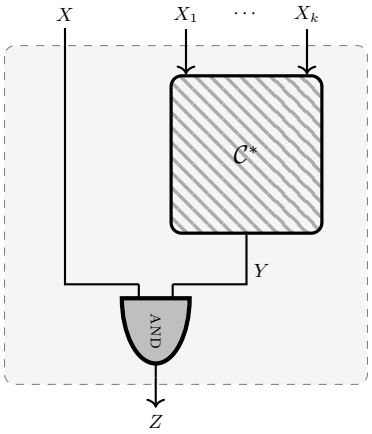


Fig. A.1: A boolean circuit computing the AND of boolean variables X and $Y \leftarrow \mathcal{C}^*(X_1, \ldots, X_k)$.

Attack-1

1. The challenger chooses uniformly at random a bit $b \in \{0,1\}$. The challenger also runs the key generation algorithm and provides the attacker with a copy of the public encryption key pk and of the evaluation key ek. Finally, the challenger maintains a state S.
2. The attacker selects a boolean circuit $\mathcal{C}^*$ and boolean inputs $X_1, \ldots, X_k$ such that $\mathcal{C}^*(X_1, \ldots, X_k) = 0$.
3. For $1 \leq j \leq k$:
   - The attacker submits the pair $(m_0, m_1) \leftarrow (X_j, X_j)$ for encryption.
   - The challenger computes and returns $c \leftarrow \mathrm{Enc}_{\mathsf{pk}}(X_j)$ to the attacker; the challenger also updates S as $S[j] \leftarrow S(X_j, X_j, c)$.
4. The attacker submits the pair $(m_0, m_1) \leftarrow (0,1)$ for encryption. The challenger returns $c \leftarrow \mathrm{Enc}_{\mathsf{pk}}(b)$ to the attacker and updates S as $S[k+1] \leftarrow (0,1,c)$.
5. The attacker defines the function

$$g_1 \colon \{0,1\}^k \to \{0,1\}, (X_1, \ldots, X_k) \mapsto \mathcal{C}^*(X_1, \ldots, X_k)$$

   and asks for the homomorphic evaluation of g_1 on state indexes $(1, \ldots, k)$. The challenger evaluates $c \leftarrow \mathrm{Eval}_{\mathsf{ek}}(g_1, S[1].c, \ldots, S[k].c)$ and returns c to the attacker. The challenger updates S as

$$S[k+2] \leftarrow \big(\mathcal{C}^*(X_1, \ldots, X_k), \mathcal{C}^*(X_1, \ldots, X_k), c\big) = (0,0,c).$$

   Recall that $Y \leftarrow \mathcal{C}^*(X_1, \ldots, X_k) = 0$ by construction; cf. Step 2.
6. The attacker defines the function

$$g_2 \colon \{0,1\} \times \{0,1\} \to \{0,1\}, (X, Y) \mapsto X \wedge Y$$

   and requests its homomorphic evaluation on state indexes $(k+1, k+2)$. The challenger computes $c \leftarrow \mathrm{Eval}_{\mathsf{ek}}(g_2, S[k+1].c, S[k+2].c)$ and returns c to the attacker. The challenger updates S as $S[k+3] \leftarrow (0 \wedge 0, 1 \wedge 0, c) = (0,0,c)$.
7. The attacker submits index $j = k+3$ for decryption. The challenger checks that $S[k+3].m_0 \stackrel{?}{=} S[k+3].m_1$ if and only if $0 \stackrel{?}{=} 0$ which is clearly satisfied. The challenger returns $Z' \leftarrow \mathrm{Dec}_{\mathsf{sk}}(S[k+3].c)$ to the attacker.
8. The attacker outputs the guess $b' \leftarrow Z'$.

Fig. A.2: IND-CPA^{D} Attack using the circuits from Figure A.1.

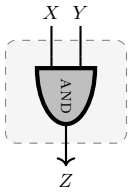


Fig. A.3: A boolean AND gate computing $Z \leftarrow X \wedge Y$.

Attack-2

1. Repeatedly calling the decryption oracle, the attacker collects T failing ciphertexts $\boldsymbol{C}^{(j)} \leftarrow (a_1^{(j)}, \ldots, a_n^{(j)}, b^{(j)})$ with $1 \leq j \leq T$;
2. The attacker computes the corresponding drift vectors

$$(\alpha_1^{(j)}, \ldots, \alpha_n^{(j)}, \beta^{(j)}) \qquad (1 \leq j \leq T) \ .$$

3. For $1 \leq i \leq n$:
   - The attacker observes the distribution of $\left\{\alpha_i^{(j)}\right\}_{1 \leq j \leq T}$.
   - If the distribution is (close to) uniform, the attacker deduces that $s_i = 0$; otherwise, that $s_i = 1$.
4. The attacker outputs $\boldsymbol{s} \leftarrow (s_1, \ldots, s_n)$.

Fig. A.4: Key Recovery Attack Based on the Circuit in Figure A.3.

Clearly, a higher value for T increases the chances of successfully recovering secret key $\boldsymbol{s}$. The authors of [15] report the recovery of 596 out of 600 secret key bits from 8434 failing ciphertexts in one of their experiments.

A.3 Analysis

We analyze the security implications of the two previous games. We consider the security goals of *(i)* indistinguishability of ciphertexts (IND) and *(ii)* security against key recovery (KR) when access to a decryption oracle is provided for honestly generated (or evaluated) ciphertexts.

IND-CPA^{D} Security: We start by evaluating the success probability of an attacker in the previous distinguishing attack in Figure A.1. We let E_1 (resp. E_2) denote the event that the ciphertext resulting from the homomorphic evaluation of g_1 (resp. g_2) is incorrect, and let p_1 (resp. p_2) denote the event probability. Writing $Y' \leftarrow \mathrm{Dec}_{\mathsf{sk}}(S[k+2].c)$ and $Z' \leftarrow \mathrm{Dec}_{\mathsf{sk}}(S[k+3].c)$, we have the following table:

E_1	E_2	Y'	Z'
`false`	`false`	0	0
`false`	`true`	0	1
`true`	`false`	1	b
`true`	`true`	1	$\neg b$

We so obtain

$$\begin{aligned}\Pr[b' = b] &= \Pr[b = Z'] \\ &= \Pr[b = 0] \cdot (1 - p_1) \cdot (1 - p_2) + \Pr[b = 1] \cdot (1 - p_1) \cdot p_2 + \\ &\qquad \Pr[b = b] \cdot p_1 \cdot (1 - p_2) + \Pr[b = \neg b] \cdot p_1 \cdot p_2 \\ &= \tfrac{1}{2} \cdot (1 - p_1) \cdot (1 - p_2) + \tfrac{1}{2} \cdot (1 - p_1) \cdot p_2 + p_1 \cdot (1 - p_2) \\ &= \tfrac{1}{2} + p_1 \cdot \left(\tfrac{1}{2} - p_2\right) \ .\end{aligned}$$

Hence, the success probability of the attacker, $\Pr[b' = b] - \frac{1}{2}$, is

$$p_1 \cdot \left(\tfrac{1}{2} - p_2\right) \approx \frac{p_1}{2}$$

since in practice $p_2 \ll \frac{1}{2}$. This success probability is negligible provided that p_1 is also negligible.

KR^{D} Security: The security notion of KR^{D} (or *security against key recovery attacks*) is related to that of IND-CPA^{D}. Formally, the security experiment of Figure 2.1 is modified by

1. restricting the encryption oracle to queries of the form (m, m);
2. requiring the adversary $\mathcal{A}$ to output a secret key sk' at the end of the attack.

The output of the experiment is defined to be 1 if $\mathsf{sk}' = \mathsf{sk}$, and 0 otherwise.

The key recovery attack as described in Figure A.4 considers a simple circuit comprising just one programmable bootstrapping. It is worth remarking that to conduct the attack, the attacker needs to know the input of a failing programmable bootstrapping.

- When circuits just contain one programmable bootstrapping, the correspondence between a failing ciphertext and a failing programmable bootstrapping is immediate. An adversary will need about $\frac{1}{p_{\mathrm{err}}}$ decryption queries to discover a failing programmable bootstrapping.
- Things are more complicated for circuits comprising multiple programmable bootstrappings. For a circuit comprising w programmable bootstrappings, the probability that at least one is failing is

  $$1 - (1 - p_{\mathrm{err}})^w \approx w \cdot p_{\mathrm{err}} \ .$$

  The key recovery attack however requires identifying exactly which programmable bootstrapping failed. If this is guessed at random, an adversary will still need about $\left(\frac{1}{w}(w \cdot p_{\mathrm{err}})\right)^{-1} = \frac{1}{p_{\mathrm{err}}}$ decryption queries to discover a failing programmable bootstrapping.

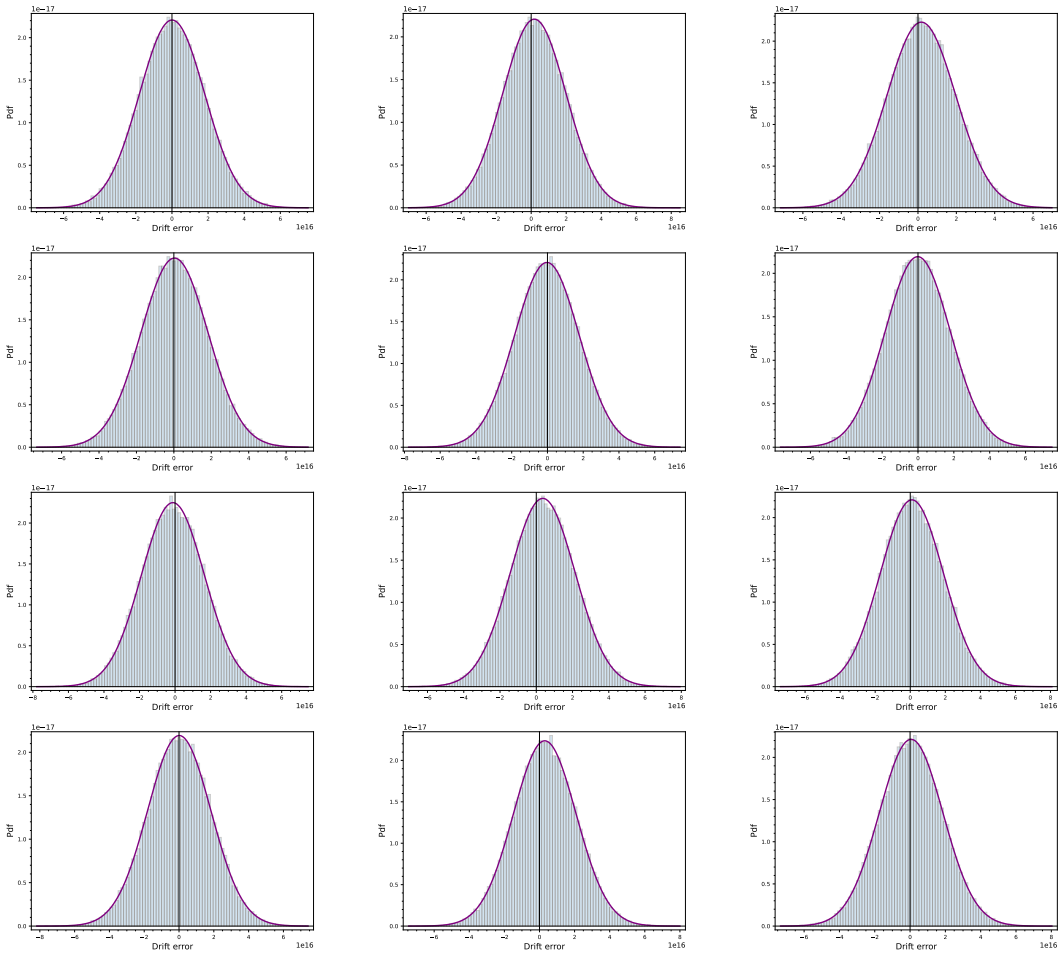
This analysis seems to indicate that limiting the number of decryption queries might constitute an effective way to prevent the recovery of the key. Unfortunately, as we show hereafter, this conclusion does not necessarily hold true.

For certain scenarios, there may be better attack strategies! Here is one. Suppose that $p_{\mathrm{err}} = 2^{-E}$. The adversary builds 'iterative' sub-circuits where the output of the previous operation serves as an input for the next one. More specifically, letting $\mathcal{M}$ the message space, consider a set of (injective) functions $\mathcal{F}_j \colon \mathcal{M} \to \mathcal{M}$ and define

$$\begin{cases} Y_1 = \mathcal{F}_1(X) \\ Y_j = \mathcal{F}_j(Y_{j-1}) \quad \text{for } 2 \leq j \leq 2^E \end{cases} .$$

Circuit $\mathcal{C}^*\colon \mathcal{M} \to \mathcal{M}$ is defined as $\mathcal{C}^* = \mathcal{F}_{2^E} \circ \mathcal{F}_{2^E-1} \circ \cdots \circ \mathcal{F}_1$. If $\mathcal{C}^*$ is evaluated homomorphically on an encryption of a message X then the resulting ciphertext corresponding to the encryption of $Z \leftarrow \mathcal{C}^*(X) = \mathcal{F}_{2^E}\Big(\mathcal{F}_{2^E-1}\big(\ldots \mathcal{F}_1(X)\big)\Big)$ has a very good chance of incorrectly decrypting—recall that $p_{\text{err}} = 2^{-E}$. Assuming that the functions $\mathcal{F}_j$ are evaluated via programmable bootstrappings, the attacker can efficiently locate the failing programmable bootstrapping through a binary search by checking whether the corresponding ciphertext is correct or not by probing the decryption oracle. In more detail, the attacker checks whether the encryption of $Y_{2^{E-1}}$ is correct. If so, the failing programmable bootstrapping occurs for some $j \in [\![2^{E-1}+1, 2^E]\!]$; if not, the failing programmable bootstrapping occurs for some $j \in [\![1, 2^{E-1}]\!]$. The attacker iterates the search on the failing intervals until there is one candidate left for j. This requires in total $E+1$ calls to the decryption oracle (the first call being used to check that the encryption of Z is indeed incorrect).

B Supplementary Experimental Data



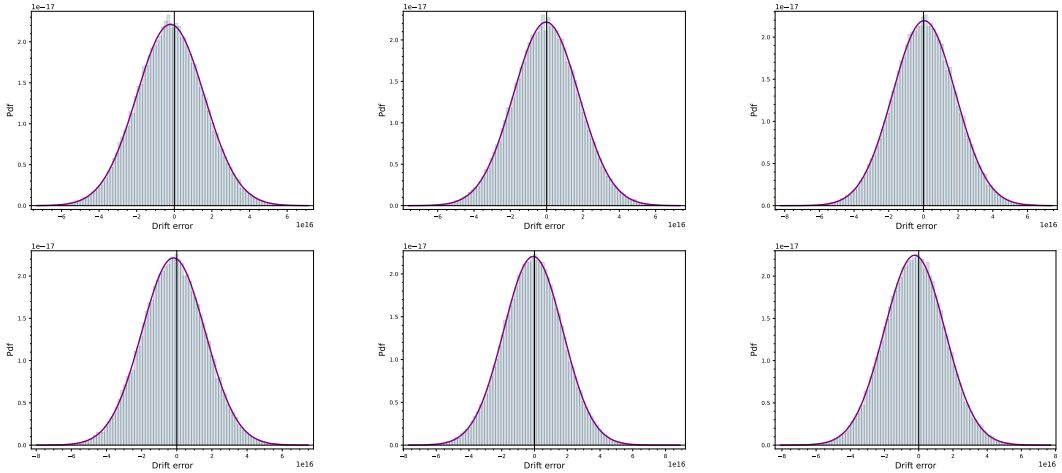


Fig. B.1: Measured vs. expected drift distribution for several drift vectors passing the quality test for parameter set $\mathfrak{Z}_{4b}^{64}$ ($T = 2^{57.76}$) and $r = 13.4$.