Achieving CPA^D security for BFV: a pragmatic approach[*]

Jean-Paul Bultel,
Marina Checri,
Caroline Fontaine,
Marc Renard,
Renaud Sirdey,
Oana Stan

Abstract Fully Homomorphic Encryption (FHE) aims at ensuring privacy of sensitive data while taking advantage of external computations and services. However, using FHE in real-world scenarios reveals new kinds of security issues. In particular, following Li&Micciancio Eurocrypt'21 seminal paper, CPA^D security has emerged as a fundamental notion for FHE, unveiling a subtle interplay between security and correctness. For *correct* (F)HE schemes, CPA security already implies CPA^D. However, all known practical FHE schemes are (R)LWE-based and, as such, are prone to decryption errors; and even if it is possible to ensure statistical correctness by selecting appropriate parameters, achieving this while maintaining malleability — the mainspring of FHE — still remains challenging. Moreover, practical CPA^D attacks have recently been designed against most known FHE schemes. We propose in this chapter a complete, simple and rigorous framework to reach CPA^D security for one of them, BFV. Our approach relies on a combination of alternate average-case/worst-case noise variance monitoring —

J.-P. Bultel
Université Paris-Saclay, CEA LIST, France, e-mail: `jean-paul.bultel@cea.fr`

M. Checri
Université Paris-Saclay, CEA LIST, France, e-mail: `marina.checri@cea.fr`

C. Fontaine
Université Paris-Saclay, ENS Paris-Saclay & CNRS, France, e-mail: `caroline.fontaine@cnrs.fr`

M. Renard
Université Paris-Saclay, CEA LIST & ENS Paris-Saclay, France, e-mail: `marc.renard@cea.fr`

R. Sirdey
Université Paris-Saclay, CEA LIST, France, e-mail: `renaud.sirdey@cea.fr`

O. Stan
Université Paris-Saclay, CEA LIST, France, e-mail: `oana.stan@cea.fr`

[*] This work was supported by the France 2030 ANR Projects ANR-22-PECY-003 SecureCompute and funded partially by the European Union under Grant Agreement No. 101167964. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Cybersecurity Competence Centre. Neither the European Union nor the European Cybersecurity Competence Centre can be held responsible for them.

based on dependencies tracking during the homomorphic calculations — and on smudging. It comes with an automated parameters setting methodology, which connects it to the recently proposed Application-Aware HE paradigm while relieving libraries end-users from the burden of enforcing the paradigm's constraints by hand.

1 Introduction

Besides intensive research to improve its computational efficiency, FHE still raises several security challenges. In particular, all practical FHE schemes such as BFV [11, 22], BGV [12], CKKS [17] and TFHE [18], "only" achieve CPA security and fail to be CCA1-secure. Specifically for FHE, Li and Micciancio 2021 seminal paper [26] introduces a slight extension of CPA security, named CPA^D, to model "passive" adversaries against FHE. In CPA^D, the adversary is granted access to a very constrained decryption oracle, which accepts only well-formed ciphertexts, i.e. fresh ciphertexts or ciphertexts derived from fresh ciphertexts by means of genuine homomorphic operations. At first glance, the intuition may be that this constrained decryption oracle does not provide more information to the adversary, as she can compute all the outputs of the decryption oracle by herself. However, in [26], Li and Micciancio have shown that this seemingly benign extension of CPA can in fact make approximate schemes such as CKKS completely insecure. Additionally, recent works [15, 16] have now shown that, beyond CKKS, "exact" schemes as BFV, BGV or TFHE, that were assumed immune to CPA^D attacks, are also CPA^D insecure, as soon as decryption errors can be made to occur with a non-negligible probability.

Moreover, some recent theoretical works, e.g. [13], have also pointed out that CPA^D security plays a foundational role for FHE security beyond CPA, and revealed a subtle interplay between scheme correctness and security. For *perfectly correct* or *statistically correct* (F)HE schemes (which respectively have a probability of decryption failure equal to zero in the perfect case, and negligible in the security parameter in the statistical case), CPA^D security is known to be implied by CPA security. Yet, as all known practical FHE schemes are (R)LWE-based, it is important to point out that even if it is possible to ensure statistical correctness in such schemes by selecting appropriate parameters, achieving this while maintaining malleability, an intrinsic property of FHE, still remains challenging. In essence, preventing decryption errors in LWE-based schemes requires a tight control of the ciphertexts noise. This is difficult to achieve without enforcing constraints on operations allowed for homomorphic evaluation or relying on the programmer's discipline in the use of FHE libraries. Another approach, having roots in leakage prevention in threshold FHE decryption protocols [2, 29], is to let decryption errors occur but prevent any leak of exploitable information even when the decryption function may output incorrect values ($\perp$ falls in this category).

Interestingly, in a recent follow-up paper dedicated to the attacks of [23, 15, 16], Alexandru et al. [1] have proposed a new security concept, referred to as the *Application Aware HE (AAHE) paradigm*, based on weaker variants of CPA^D security, that focuses on specific circuit classes and noise estimation strategies. The authors then argue that these variants are more appropriate in practice. They also provide guidelines for implementing this approach, but leave the responsibility of enforcing circuit constraints or appropriate usage of a noise estimation strategy to the library users.

In the present chapter, we propose a complete, simple and consistent framework to achieve CPA^D security for BFV [22] under relaxed correctness. It is built on a combination of alternate average-case/worst-case noise variance monitoring and on the addition of smudging noise of appropriately large variance, at decryption. This variance depends on both the security parameters and the ciphertext noise variance tracked during the homomorphic calculations. The framework comes with a methodology for an automatic choice of parameters, avoiding the pitfall of a secure but useless scheme (because of too frequent decryption errors or $\perp$ answers).

In addition, our approach proposes an *automated* way to achieve AAHE security, i.e. achieving CPA^D security for somewhat correct FHE schemes (focusing on BFV) relatively to a given circuit or class of circuits with an automatic adaptive noise variance bounding strategy. For each homomorphic operation, it combines several existing techniques (e.g., average/worst-case variance bounding or smudging) and takes into account the potential dependencies in their ciphertexts inputs.

Finally, since CPA^D and AAHE security notions essentially model passive adversaries and thus do not go beyond a honest-but-curious threat model, we also investigate a more advanced three party setting where a *honest-but-curious* server performs homomorphic computations over data from a *malicious (active)* provider, then returns the results to a client, owner of the decryption key.

Contributions summary

The main goal of this work is to propose a complete, simple and consistent framework to achieve CPA^D security for BFV while allowing an automatic, more flexible but still reliable, choice of parameters. Our contributions are summarized below.

- We design CPA^D-secure HE schemes based on the BFV popular cryptosystem. More precisely, we construct several variants of that scheme and we prove that they are all CPA^D secure. We achieve this by the combination of two techniques: monitoring of the ciphertext noise variance based on the tracking of dependencies during an homomorphic calculation (either worst case or average case) and the well-known technique of smudging.
- We automatize the Application Aware HE (AAHE) framework in this context and provide a concrete instantiation of AAHE for BFV with an associated methodology for the automatic setting of parameters to ensure reliable decryption. We investigate the noise variance for fresh BFV ciphertexts and give refined bounds on the ciphertext noise variances in the worst and average cases.
- We also investigate a three (non-colluding) parties setting in which an honest-but-curious server performs homomorphic evaluations over encrypted data from a possibly malicious provider (owner of the data to encrypt) for a client with decryption capabilities. In this context, we propose an approach where the data provider uses PRF-based encryption towards the server, and shows that this framework can easily

achieve independence of the input ciphertext noises. Because, in this protocol, the provider does not build by itself or even see any FHE ciphertexts, it is not anymore an adversary against the FHE scheme security.
- Finally ,we validate our approach by taking into account a “noise budget” and a circuit class to allow a fine-tuned choice of BFV parameters. We show how to select a set of parameters to evaluate a circuit depending on chosen security and reliability levels. We then give illustrative timings for several parameter sets over a representative corpus of algorithms.

Outline of the paper

We first recall some background definitions and results in Section 2, and then discuss related works in Section 3. Then, we focus on our contributions: we design in Section 4 variants of BFV which achieve CPA^D security; we propose in Section 5 a methodology to set their parameters; we subsequently describe our PRF-based encryption approach in Section 6. Finally, Section 7 is devoted to experiments. We conclude and discuss future work in Section 9.

2 Background

Given $l, u \in \mathbb{Z}$, we use $[\![l, u]\!]$ to denote the set $\{l, l+1, ..., u-1, u\}$. Reduction modulo q is denoted as $[.]_q$. We use this latter notation explicitly only when it avoids possible ambiguities.

2.1 Basic definitions

We define an encryption scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ over key space $\mathcal{K}$ with plaintext domain $\mathcal{P}$ and ciphertext domain $\mathcal{C}$ (with COIN denoting the randomness space of $\mathcal{E}.\mathsf{Enc}$), as a triplet of PPT algorithms:

- KeyGen: on input 1^λ, outputs an encryption key ek and a decryption key sk, with $(\mathsf{ek}, \mathsf{sk}) \in \mathcal{K}$.
- $\mathsf{Enc}_{\mathsf{ek}}$: on input $m \in \mathcal{P}$ and the encryption key ek, outputs an encryption $c \in \mathcal{C}$ of m. We will sometimes *externalize* the randomness used in the encryption function by means of the notation $\mathsf{Enc}_{\mathsf{ek}}(m; r)$, with $m \in \mathcal{P}$ and $r \in \mathsf{COIN}$ (in this case, the function $\mathsf{Enc}_{\mathsf{ek}} : \mathcal{P} \times \mathsf{COIN} \longrightarrow \mathcal{C}$ is deterministic).
- $\mathsf{Dec}_{\mathsf{sk}}$: on input $c \in \mathcal{C}$ and the decryption key sk, outputs a decryption $m \in \mathcal{P} \cup \{\perp\}$ of c.

When ek is public, we say that $\mathcal{E}$ is a *public-key* encryption scheme and use pk to denote ek. When for all $(\mathsf{ek}, \mathsf{sk}) \in \mathcal{K}$ and all $m \in \mathcal{P}$ we have that

$$\Pr_{r \in \mathsf{COIN}}(\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{ek}}(m;r)) \neq m) \leq \mathrm{neg}(\lambda), \tag{1}$$

we say that $\mathcal{E}$ is *correct*. When the above probability is equal to 0, we will talk of *perfect* correctness. If there is no ambiguity, we omit the ek and sk subscripts from Enc and Dec.

Given a function class $\mathcal{F}_H$, we define $\mathcal{E}_H$ a *homomorphic encryption* (HE) scheme as an encryption scheme augmented by a *deterministic*[2] polynomial-time algorithm Eval which, on input function $f \in \mathcal{F}_H$ of arity K and ciphertexts $c_1, ..., c_K \in \mathcal{C}$, outputs a new *evaluated* ciphertext. When $\mathcal{E}_H$ satisfies condition (1) and when Eval is such that, for all $(\mathsf{ek}, \mathsf{sk}) \in \mathcal{K}$, all functions $f \in \mathcal{F}_H$ and all messages $m_1, ..., m_K \in \mathcal{P}$

$$\begin{aligned}\Pr_{\vec{r} \in \mathsf{COIN}^K}(&\mathsf{Dec}(\mathsf{Eval}(f, \mathsf{Enc}(m_1;r_1), ..., \mathsf{Enc}(m_K;r_K))) \\ &\neq f(m_1, ..., m_K)) \leq \mathrm{neg}(\lambda),\end{aligned} \tag{2}$$

we say that $\mathcal{E}_H$ is a *statistically correct* or, more simply, a *correct* HE scheme. When it is not the case, we say that $\mathcal{E}_H$ is an *approximate* HE scheme. Consistently with [27], to avoid arbitrary schemes with unreliable Eval to be labelled as approximate HE schemes, we add an additional condition that, for some (small) $\varepsilon \geq 0$, the following holds

$$\begin{aligned}\Pr_{\vec{r} \in \mathsf{COIN}^K}(&||\mathsf{Dec}(\mathsf{Eval}(f, \mathsf{Enc}(m_1;r_1), ..., \mathsf{Enc}(m_K;r_K))) \\ &- f(m_1, ..., m_K)||_\infty \leq \varepsilon) \geq \mu,\end{aligned} \tag{3}$$

with[3] $\mu \geq \frac{3}{4}$ (as an arbitrary value that is non-negligibly above 1/2). Lastly, a scheme such that $\varepsilon = 0$ and $\frac{3}{4} \leq \mu < 1 - \mathrm{neg}(\lambda)$ is said to be *somewhat correct*.

2.2 Security notions

2.2.1 CPA^D security.

The CPA^D game has been introduced in the context of approximate FHE [26]. CPA^D security is a slight extension of CPA security defined by the following Left-Or-Right multiple challenge security game.

Given a homomorphic encryption scheme $\mathcal{E}_H = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$, an adversary $\mathcal{A}$ and a value λ for the security parameter, the game is parameterized by a bit $b \overset{\$}{\leftarrow} \{0,1\}$, unknown to $\mathcal{A}$, and an initially empty state S of message-message-ciphertext triplets:

- Key generation: Run $(\mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and give ek to $\mathcal{A}$ (when the scheme is public-key).
- Encryption request: When $\mathcal{A}$ queries $(\texttt{plaintext}, m)$, with $m \in \mathcal{P}$, compute $c = \mathsf{Enc}(m)$, give c to $\mathcal{A}$ and do $S := [S; (m, m, c)]$.
- Challenge request: When $\mathcal{A}$ queries $(\texttt{test\_messages}, m_0, m_1)$, with $m_0, m_1 \in \mathcal{P}^2$ and $m_0 \neq m_1$, compute $c = \mathsf{Enc}(m_b)$, give c to $\mathcal{A}$ and do

[2] As it is the case for mainstream FHE schemes like BFV, BGV, TFHE and even CKKS.

[3] In practice μ is typically chosen above $1 - 2^{-40}$. In some contexts, e.g. [1], μ even has to be at least $1 - \mathrm{neg}(\lambda)$.

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

- Evaluation request: When $\mathcal{A}$ queries $(\texttt{eval}, f, l_1, \ldots, l_K)$ $(l_i \in \{1, \ldots, |S|\}, \forall i)$, compute
$$\begin{cases} m'_0 = f(S[l_1].m_0, \ldots, S[l_K].m_0), \\ m'_1 = f(S[l_1].m_1, \ldots, S[l_K].m_1), \\ c' \ \ = \mathsf{Eval}(f, S[l_1].c, \ldots, S[l_K].c), \end{cases}$$
then give c' to $\mathcal{A}$ and update $S := [S; (m'_0, m'_1, c')]$.
- Decryption request: When $\mathcal{A}$ queries $(\texttt{ciphertext}, l)$ $(l \in \{1, \ldots, |S|\})$ proceed as follows: if $S[l].m_0 \neq S[l].m_1$ then return $\perp$ to $\mathcal{A}$, otherwise return her $\mathsf{Dec}(S[l].c)$.
- Guessing stage (after a polynomial number of interleaved encryption and decryption requests): When $\mathcal{A}$ outputs $(\texttt{guess}, b')$, the game ends as follows. If $b' = b$ then $\mathcal{A}$ wins the game. Otherwise, $\mathcal{A}$ loses the game.

Let us emphasize several points concerning the CPA^D game. First, the decryption oracle accepts only ciphertexts from the game state which are necessarily *well-formed*: i.e. either ciphertexts produced by an encryption or challenge request, or derived by the evaluation oracle via an evaluation request, i.e. derived by correctly applying homomorphic operators to well-formed ciphertexts. As such, the above game does not capture any CCA aspects. Second, note that for *FHE schemes not satisfying the correctness assumption*, even when $S[l].m_0 = S[l].m_1$, we may have $\mathsf{Dec}(S[l].c) \neq S[l].m_0$ and $\mathsf{Dec}(S[l].c) \neq S[l].m_1$. Thus, for such schemes, the decryption oracle grants $\mathcal{A}$ access to information she cannot compute on her own, possibly resulting in a guessing advantage depending on whether or not the cryptosystem at hand is CPA^D secure. Third, recall that in the CPA^D game $\mathcal{A}$ controls the homomorphic calculations performed, as f is included in the evaluation request.

2.2.2 CPA and CCA1 security.

Recall that, by convention, CPA and CCA1 security notions are usually single challenge. In Appendix 10, we describe them more in details and define them relatively to the CPA^D game of the previous section.

In the general regime where approximate or somewhat correct FHE schemes are allowed, we also have the following separation [13]: CPA $\prec$ CPA$^D_1$ $\prec$ CCA1, where CPA$^D_1$ is the restriction of CPA^D to the single challenge case where the adversary is allowed only one request of the form $(\texttt{test\_messages}, m_0, m_1)$ with $m_0 \neq m_1$ and with the decryption oracle closing after the unique challenge request (similar in spirit to CCA1).

2.3 BFV

BFV [11, 22] is a RLWE-based homomorphic scheme defined as follows. Let n, a power of 2, be the *polynomial degree*, q be the *ciphertext modulus*, t be the *plaintext modulus*,

$\Delta = \lfloor \frac{q}{t} \rfloor$, χ_s be the distribution of the secret key (usually, the secret key distribution is a binary or ternary uniform distribution) and χ be the error (discrete Gaussian) distribution over $\mathbb{Z}_q$. For $k \in \mathbb{N}$, we note $R_k = {}^{(\mathbb{Z}/k\mathbb{Z})[X]}\!/\!_{(X^n+1)}$.

- $\mathsf{BFV.KeyGen}$: sample $s \overset{\$}{\leftarrow} \chi_s$, $a \overset{\$}{\leftarrow} R_q$, $e \overset{\$}{\leftarrow} \chi$. Set the secret and public-keys as $\mathsf{sk} = s$ and $\mathsf{pk} = \left(a, b = [-a \cdot s + e]_q\right)$.
- $\mathsf{BFV.Enc}(\mathsf{pk} = (\mathsf{pk}.a, \mathsf{pk}.b), m \in R_t)$: sample $u \overset{\$}{\leftarrow} \chi_s$, $e_1, e_2 \overset{\$}{\leftarrow} \chi$, and return $\mathbf{c} = \left([\mathsf{pk}.a \cdot u + e_2]_q, [\Delta \cdot m + \mathsf{pk}.b \cdot u + e_1]_q\right)$.
- $\mathsf{BFV.Dec}(\mathsf{sk}, \mathbf{c} = (c_0, c_1))$: compute and return $\left[\left\lfloor \frac{t}{q} \cdot [c_1 + c_0 \cdot \mathsf{sk}]_q \right\rceil\right]_t$.

We refer the reader to Appendix 11 for a complete description of the scheme, including its homomorphic operators.

2.4 Smudging

Smudging is a technique that consists in "hiding" a small noise by flooding it with a much larger noise such that the effect of the small noise becomes negligible. Smudging was first introduced in [5] in the context of threshold PKE and later in [2, 9] in the context of threshold FHE. For threshold FHE it was designed such that a decryption oracle over well-formed ciphertexts can be simulated without actually decrypting in the relevant security reductions and has been used since then in several constructions, e.g. [29]. Beyond theshold FHE, smudging has also been suggested as a countermeasure to CPA^D attacks against CKKS [27] (where it is referred to as noise flooding) or the other "exact" FHE [15]. Smudging comes into different flavors depending on whether the statistical distance or the Rényi divergence is considered [19, 10, 28] or whether worst-case/non-worst-case smudging should be performed [6, 23].

In this work, we consider *worst-case* smudging based on the *statistical distance*. Recall that the usual statistical distance for two discrete random variables X and Y is defined as $d(f_X, f_Y) = \frac{1}{2}\sum_{k=-\infty}^{+\infty} |P(X = k) - P(Y = k)|$. We then rely only on the simple "Smudging Lemma" [2, Lemma 1], which we reproduce below.

Lemma 1 *[2, Lemma 1] Let e_1 and e_2 be two positive integers and let $\mathrm{e} \in [\![-\mathrm{e}_1, \mathrm{e}_1]\!]$ be a fixed integer. Let e' be chosen uniformly at random in $[\![-\mathrm{e}_2, \mathrm{e}_2]\!]$. Then, if $\mathrm{e}_2 \geq 2^\lambda \mathrm{e}_1$ the statistical distance between the distribution of e' and that of $\mathrm{e} + \mathrm{e}'$, $d(f_{\mathrm{e}'}, f_{\mathrm{e}'+\mathrm{e}})$, is bounded by neg$(\lambda)$.*

We now prove two simple lemmas required to be able to smudge any centered random variable based on its variance.

Lemma 2 *Let ε be a centered random variable with variance σ_{ct}^2. Further let $e_1 = \sigma_{\mathrm{ct}}\sqrt{2^\lambda}$, then $P(\varepsilon \notin [\![-e_1, e_1]\!]) \leq 2^{-\lambda}$.*

Proof. Recall that the Chebychev bound tells that $P(|\varepsilon| \geq B) \leq \frac{\sigma_{\mathrm{ct}}^2}{B^2}$. Now, choosing e_1 such that $\frac{\sigma_{\mathrm{ct}}^2}{{e_1}^2} = 2^{-\lambda}$, i.e. $e_1 = \sigma_{\mathrm{ct}}\sqrt{2^\lambda}$, yields the claim. □

Remark that we used the Chebychev bound in the above proof (rather than e.g. the much sharper Chernoff bound) as it is distribution independent. This distribution independence assumption is needed as, although the noise distribution in fresh ciphertexts is (discrete) Gaussian, the noise distribution in an evaluated ciphertext after an arbitrary sequence of homomorphic operations is generally unknown.

If we now choose $e_2 = 2^{\lambda} e_1 = 2^{\frac{3\lambda}{2}} \sigma_{\text{ct}}$, then the Smudging Lemma (Lemma 1) applies, directly leading to the following Lemma.

Lemma 3 *Let ε be a centered random variable with variance σ_{ct}^2 and let $e_2 = 2^{\frac{3\lambda}{2}} \sigma_{\text{ct}}$. Then, $d(f_e, f_{e+\varepsilon}) \leq 2^{-\lambda}$, where e is picked uniformly in $[\![-e_2, e_2]\!]$.*

We do not claim that this simple approach is optimal and more advanced approaches [30] may yield smaller smudging noise bounds or variances, eventually leading to smaller LWE parameters, thus better performances. Yet, as will be illustrated in Section 7, the present approach already allows to obtain practical performances.

3 Related Works

3.1 CPA^D security and attacks

The CPA^D security notion has been introduced in 2021 by Li and Micciancio [26] to model passive adversaries against FHE. It is an extension of CPA security, where the adversary is granted access to an evaluation oracle and a very constrained decryption oracle. At first glance, the intuition may be that this constrained decryption oracle does not provide more information to the adversary, as she can compute all the outputs of the decryption oracle by herself. However, Li and Micciancio demonstrated that these intuitions are erroneous for approximate FHE schemes such as CKKS, for which it turns out that the differences $\mathsf{Dec}(\mathsf{Enc}(m)) - m$ or $\mathsf{Dec}(\mathsf{Eval}(f, \mathsf{Enc}(m))) - f(m)$ leak the LWE noises present in the ciphertexts, resulting in the ability for the adversary to recover the secret decryption key of the scheme. In [27], the authors then proposed a practical countermeasure by *smudging* the LWE noise of a ciphertext during the decryption process so that the attacker can extract no information from the decryption of such a ciphertext. But in 2024, Guo et al. [23] proposed a CPA^D attack on the CKKS scheme with noise flooding, highlighting that an attack was still possible when the smudging technique was based on non-worst-case noise estimation.

The same year, [16] and [15] introduced new attacks that demonstrated that CPA^D security could pose serious challenges for somewhat correct schemes as well, thus showing that CPA^D security is not just a theoretical but a real threat in practice for all FHE schemes. Cheon et al. [16] proposed CPA^D attacks against somewhat correct schemes. They described a BGV/BFV CPA^D attack based on migrating the noise polynomial into the plaintext domain to recover the noise. Then, they explored a CPA^D attack on the TFHE scheme by taking advantage of bootstrapping errors. Checri et al. [15] investigated a CPA^D attack against all somewhat correct schemes, relying on the linear part of these schemes. They proposed to control the noise growth in some encryptions, triggering incorrect decryption to gain information on the secret LWE noise of the ciphertext and, eventually, recovering the secret decryption key. They ended their article by initiating a discussion towards concrete countermeasures.

3.2 The Application Aware HE paradigm

Following these attacks, Alexandru et al. [1] proposed a new security paradigm based on weaker variants of CPA^D security, namely the Application-Aware HE (AAHE) paradigm.

In essence, this new definition acknowledges that for somewhat correct FHE schemes, CPA^D security should be defined relative to a circuit class and a noise estimation strategy rather than absolutely. So the cryptosystem parameters should be set relative to these, and the homomorphic evaluations should (somehow) be limited to the functions or circuits in the class. For somewhat correct schemes, allowable functions or circuits can be defined by a noise standard deviation budget, which should not be exceeded. However, for approximate schemes, the issue of meaningfully defining application-aware security is much more subtle, and the majority of [1] is devoted to tackling this. Additionally, the application-aware framework is not relevant for “true” FHE when the parameters are chosen to achieve statistical correctness over fresh ciphertexts and so that bootstrapping errors occur only with a probability negligible in the security parameter.

The AAHE paradigm acknowledges the issues created by the different CPA^D attacks, and provides guidelines for users in order to mitigate them. The only drawback is that the burden of enforcing the above constraints lies, so far, solely on the library user’s shoulders. However, it is presently unclear how to automatically enforce the AAHE paradigm, for example when the average-case noise estimation strategy is used. Therefore, it is an interesting line of research to attempt to improve on this first approach towards more robust and automated mitigation approaches. This is, to some extent, the goal that we achieve in this chapter.

3.3 Strong CPA^D

In 2024, Bernard et al. [7] introduced a stronger variant of CPA^D, called strong CPA^D (sCPA^D). In this model, the adversary has the ability to control the randomness used in the encryption function while remaining honest-but-curious. Bernard et al. showed that sCPA^D is strictly stronger than CPA^D, for both public-key and private-key settings. They introduced a new correctness notion named ACER (Adversarially Chosen Encryption Randomness), and showed that CPA security and ACER correctness imply sCPA^D security. They finally succeeded in instantiating concrete sCPA^D schemes.

Unlike [7], we focus on BFV in levelled-mode rather than on bootstrapped schemes. In section 6, we further deal with the issue of adversarially-chosen randomness in a different way from [7], at the protocol level, by ensuring that the adversary (the data provider in that setup) neither builds on its own nor sees any FHE ciphertext.

4 CPA^D secure extensions of BFV

In this section, we investigate a CPA^D secure extension of BFV.

The CPA^D threat model assumes a passive adversary with access to an encryption oracle, an evaluation oracle and a constrained decryption oracle. In other words, the CPA^D game assumes that all entities involved in a protocol follow the cryptosystem specifications. By *independent BFV ciphertexts*, we denote fresh ciphertexts which are well-formed and independently generated with the same key material. However, as we will show, the noises in two *independent BFV ciphertexts* may still exhibit some dependencies.

4.1 Noise covariance in and between fresh BFV ciphertexts

In Appendix 12, we provide detailed covariance calculations regarding the noise in fresh BFV ciphertexts. We first check that the covariance between the noises in the same coefficient of two distinct fresh well-formed BFV ciphertexts is zero (despite of their dependency to the public-key material). This essentially establishes that these noises are uncorrelated and means that we can safely apply an average case variance monitoring strategy (as $\text{Var}[X+Y] = \text{Var}[X] + \text{Var}[Y]$ is also true for uncorrelated random variables) when two distinct fresh well-formed BFV ciphertexts go through some homomorphic operations. We do so for both the conventional BFV public-key as well as for the variant with a Regev-style public-key[4], as both cases are interesting in this work. In both cases, we also study the covariance between noises in distinct coefficients of the same well-formed ciphertext. This time the covariance is not necessarily 0.

Because the calculations are particularly tedious (though not conceptually difficult), we provide them only in Appendix 12 but then assume the following facts (which are valid regardless of whether the standard or a Regev-style public-key pattern is used).

Fact 1. *Let (a, b) and (a', b') denote two independent fresh BFV ciphertexts (generated using the public-key) and let e and e' denote their respective noise polynomials. Then for all $i, j \in [\![0, n-1]\!]$, $e_{|i}$ and $e'_{|j}$ (with e_i being the i-th coefficient of e) are* dependent *random variables.*

Fact 2. *Let (a, b) and (a', b') denote two independent fresh BFV ciphertexts (generated using the public-key) and let e and e' denote their respective noise polynomials, then for all $i, j \in [\![0, n-1]\!]$, $Cov(e_{|i}, e'_{|j}) = 0$.*

Fact 3. *Let (a, b) denote a fresh BFV ciphertext (generated using the public-key) and let e denote its noise polynomial, then for all $i, j \in [\![0, n-1]\!]$, $e_{|i}$ and $e_{|j}$ ($i \neq j$) are* dependent *random variables.*

[4] I.e. the public key is generated by picking a random set of encryptions of zero and summing random subsets of them at encryption.

Fact 4. *Let (a, b) denote a fresh BFV ciphertext (generated using the public-key) and let e denote its noise polynomial, then for all $i, j \in [\![0, n-1]\!]$, $Cov(e_{|i}, e_{|j}) \neq 0$ (with overwhelming probability).*

4.2 Abstract noise variance tracking mechanics

We start from the BFV scheme and build the scheme $\mathcal{E}_H$ in this section. First $\mathcal{E}_H.\mathcal{P} = \mathsf{BFV}.\mathcal{P} \cup \{\perp\}$ and $\mathcal{E}_H.\mathcal{C} = \mathsf{BFV}.\mathcal{C} \times \mathbb{R}_+ \times 2^{\mathbb{N}}$ with the semantic that $\mathcal{E}_H$'s ciphertexts are triplets (c, σ^2, τ) where c is a BFV ciphertext, σ^2 is an upper bound for the variance of the noise in that ciphertext and τ is a set of integer values, which is later specified (basically it allows to track ciphertexts dependencies, during a homomorphic calculation). We denote σ_0^2 the noise variance in a fresh, well-formed BFV ciphertext. Let $\mathcal{O}_1$ and $\mathcal{O}_2$ respectively denote the sets of atomic univariate and bivariate operations supported by BFV. That is, $\mathcal{O}_1$ consists of additions by constants, multiplications by constants as well as slot rotations. Also, $\mathcal{O}_2$ consists of addition and multiplication of two ciphertexts. Then, we assume given the functions:

- $\mathsf{B}_1 : \mathcal{O}_1 \times \mathbb{R}_+ \longrightarrow \mathbb{R}_+$ such that when operation $\omega \in \mathcal{O}_1$ is applied to a ciphertext with noise variance at most σ_{ct}^2, the noise variance in the output ciphertext is bounded by $\mathsf{B}_1(\omega, \sigma_{\mathsf{ct}}^2)$. Remark that for univariate operations, as only one ciphertext is involved, there is no issue due to noise dependencies[5].
- $\mathsf{B}_2^{\text{AvgC}} : \mathcal{O}_2 \times \mathbb{R}_+^2 \longrightarrow \mathbb{R}_+$ such that when $\omega \in \mathcal{O}_2$ is applied over two ciphertexts with *uncorrelated* noises of respective variances at most $\sigma_{\mathsf{ct}_1}^2$ and $\sigma_{\mathsf{ct}_2}^2$, the noise variance in the output ciphertext is bounded by $\mathsf{B}_2^{\text{AvgC}}(\omega, \sigma_{\mathsf{ct}_1}^2, \sigma_{\mathsf{ct}_2}^2)$.
- $\mathsf{B}_2^{\text{WstC}} : \mathcal{O}_2 \times \mathbb{R}_+^2 \longrightarrow \mathbb{R}_+$ such that when $\omega \in \mathcal{O}_2$ is applied over two ciphertexts with *correlated* noises of respective variances at most $\sigma_{\mathsf{ct}_1}^2$ and $\sigma_{\mathsf{ct}_2}^2$, the noise variance in the output ciphertext is bounded by $\mathsf{B}_2^{\text{WstC}}(\omega, \sigma_{\mathsf{ct}_1}^2, \sigma_{\mathsf{ct}_2}^2)$.

4.3 Illustrative noise formulas

The present approach is agnostic to the noise formula used as long as they satisfy the properties of the previous section. However, note that for bivariate operations, we use different formulas for the Average Case and Worst Case noise formulas depending on whether or not the noises in the involved ciphertexts are correlated. As a simple example, for the homomorphic addition operator we have

$$\mathsf{B}_2^{\text{AvgC}}(\mathsf{Add}, \sigma_{\mathsf{ct}_1}^2, \sigma_{\mathsf{ct}_2}^2) = \sigma_{\mathsf{ct}_1}^2 + \sigma_{\mathsf{ct}_2}^2,$$

[5] Note that, to handle bivariate operations that take a *unique* ciphertext and a plaintext as parameters (e.g. a multiply-by-α), we consider that there is one univariate element of $\mathcal{O}_1$ per possible plaintext parameter (and the associated noise variance bound calculation which depends on the constant value). In other words, there is no multiply-by-α operation, but multiply-by-0, multiply-by-1, ..., multiply-by-$t-1$ (this is just a convention).

and[6]

$$\mathsf{B}_2^{\text{WstC}}(\mathsf{Add}, \sigma^2_{\mathsf{ct}_1}, \sigma^2_{\mathsf{ct}_2}) = 4\max(\sigma^2_{\mathsf{ct}_1}, \sigma^2_{\mathsf{ct}_2}).$$

For the multiplication of two ciphertexts ct_1 and ct_2, with the second one over ciphertext modulus $q_{l'}$ and with noise polynomials of variances v and respectively v' in each coefficient we propose here the formula for worst-case variance per coefficient adapted from [8] and derived in Appendix 13:

$$\begin{aligned}\mathsf{B}_2^{\text{WstC}}(\mathsf{Mul}, \sigma^2_{\mathsf{ct}_1}, \sigma^2_{\mathsf{ct}_2}) = \frac{t^2 n^2 v_s}{12}\Big(& v f(T_1+1) + v' f(T_2+1) \\ &+2\sqrt{vv' f(T_1+1) f(T_2+1)}\Big) + \frac{t^2}{q_{l'}^2} 2n^2 v^2 v'^2\end{aligned} \tag{4}$$

with $f(\iota) = -e^{\alpha - \beta\iota - \gamma\iota^2} + \delta$. The function f, defined in [8], was found with a heuristic method with α, β, γ and δ depending on the distribution of χ_s and the ring dimension n.

For completeness, we also provide concrete examples of variances for the average case from [8] which we summarize in Table 1 (see also App. 13).

Let us also note that most approaches in the literature are proposing worst-case bounds for the noise in BFV ciphertexts based on the infinity norm, e.g. [25], or the canonical norm [21, 20, 24]) and not on the noise variance.

Homomorphic operation	Variance
Enc	$\frac{t^2}{q^2}\left(\frac{1}{12} + nv_e v_u + v_e + nv_e v_s\right)$
$\mathsf{Add}(\mathsf{ct}_1, \mathsf{ct}_2)$	$v_1 + v_2$
$\mathsf{MultConst}(\alpha, \mathsf{ct}_1)$	$\frac{(t^2-1)n}{12} v_1$
$\mathsf{Mult}(\mathsf{ct}_1, \mathsf{ct}_2)$	$\frac{t^2 n^2 v_s}{12}\left(v_1 f(T_1+1) + v_2 f(T_2+1)\right)$
$\mathsf{Relin}^{\text{GHS}}(\mathsf{rlk}, p, \mathsf{ct}_1)$	$v_1 + \frac{t^2}{12q^2}(nv_e + 1 + nv_s)$

Table 1 Average-case bounds on the ciphertext noise variance depending on the homomorphic operation, where v_1 and v_2 are the error coefficients' variances of independently-computed ciphertexts ct_1 and ct_2 modulo q, T_1 and T_2 are the degrees of errors as polynomials in s of ct_1, ct_2, and f is the function $f(\iota) = -e^{\alpha-\beta\iota-\gamma\iota^2} + \delta$ defined in [8]. Also, v_s, v_e and v_u respectively denote the variances of a coefficient of the secret key s, of the error e and of the u polynomial used in BFV encryption.

4.4 A new CPA^D secure variant: $\mathcal{E}_H$

When evaluating a circuit $\mathcal{C}$, the FHE computer must keep track of the ciphertext noise standard deviations. In order to bound it correctly, it must then handle ciphertexts dependencies at each step of the calculation. If the input ciphertexts of a bivariate operation

[6] Given two random variables X and Y, $\mathrm{Var}[X+Y] = \mathrm{Var}[X] + \mathrm{Var}[Y] + 2\mathrm{Cov}(X,Y) \leq 4\max(\mathrm{Var}[X], \mathrm{Var}[Y])$.

are independent, the server can simply use the Average-Case noise estimation strategy. But if the two inputs ciphertexts are dependent, i.e. when their noises are correlated, using an Average-Case bound may result in a significant underestimation of the variance and a Worst-Case strategy should be used. To detect such ciphertexts dependencies, we use a set τ of identifiers pointing to antecedent ciphertexts, and initialized with a unique identifier for each input ciphertext. Hereunder, we define a new scheme $\mathcal{E}_H$, which will eventually achieve CPA^D security via such means.

- $\mathcal{E}_H$.KeyGen: runs BFV.KeyGen to get BFV.pk and BFV.sk.
- $\mathcal{E}_H$.Enc: given $m \in \mathcal{E}_H.\mathcal{P}$, generate a unique identifier id $\in \mathbb{N}$ and return $\mathsf{ct} = (\mathsf{ct}.c, \mathsf{ct}.\sigma^2, \mathsf{ct}.\tau) = (\mathsf{BFV.Enc}(m), \sigma_0^2, \{\mathsf{id}\})$, with σ_0^2 the variance of the coefficients in the fresh noise polynomial inside $\mathsf{ct}.c$.
- $\mathcal{E}_H$.Eval (univariate op.): given $\omega \in \mathcal{O}_1$ and $\mathsf{ct} \in \mathcal{E}_H.\mathcal{C}$, return

$$\Big(\mathsf{BFV.Eval}(\omega, \mathsf{ct}.c), \mathsf{B}_1(\omega, \mathsf{ct}.\sigma^2), \mathsf{ct}.\tau\Big).$$

- $\mathcal{E}_H$.Eval (bivariate op.): given $\omega \in \mathcal{O}_2$ and $\mathsf{ct}_1, \mathsf{ct}_2 \in \mathcal{E}_H.\mathcal{C}^2$, when $\mathsf{ct}_1.\tau \cap \mathsf{ct}_2.\tau = \emptyset$ return

$$\Big(\mathsf{BFV.Eval}(\omega, \mathsf{ct}_1.c, \mathsf{ct}_2.c), \mathsf{B}_2^{\text{AvgC}}(\omega, \mathsf{ct}_1.\sigma^2, \mathsf{ct}_2.\sigma^2), \mathsf{ct}_1.\tau \cup \mathsf{ct}_2.\tau\Big). \quad (5)$$

  Otherwise, $\mathsf{ct}_1.\tau \cap \mathsf{ct}_2.\tau \neq \emptyset$. In this latter case, return

$$\Big(\mathsf{BFV.Eval}(\omega, \mathsf{ct}_1.c, \mathsf{ct}_2.c), \mathsf{B}_2^{\text{WstC}}(\omega, \mathsf{ct}_1.\sigma^2, \mathsf{ct}_2.\sigma^2), \mathsf{ct}_1.\tau \cup \mathsf{ct}_2.\tau\Big). \quad (6)$$

- $\mathcal{E}_H$.Eval ($f \in \mathcal{F}$): given $\mathsf{ct}_1, ..., \mathsf{ct}_K$ where K is the arity of f, this evaluation is performed by running the sequence of single op. Eval corresponding to the unitary operations making up f.
- $\mathcal{E}_H$.Dec: given $\mathsf{ct} = (c, \sigma^2, \tau) \in \mathcal{E}_H.\mathcal{C}$, draw a smudging noise polynomial e_{smg} uniformly from $[\![-2^{\frac{3\lambda}{2}}\sigma, 2^{\frac{3\lambda}{2}}\sigma]\!]^n$, where $\sigma = \sqrt{\mathsf{ct}.\sigma^2}$.
  Then return $\perp$ when $||e_{\text{smg}}||_\infty \geq \frac{\Delta}{2}$ and $\mathsf{BFV.Dec}(\mathsf{ct}.c + (0, e_{\text{smg}}))$ otherwise (with, recall, $\Delta = \lfloor \frac{q}{t} \rfloor$).

By construction of $\mathcal{E}_H$, we have the following Lemma.

Lemma 4 *Let $ct = \mathcal{E}_H.Eval(f, ct_1, ..., ct_K)$, and denote by σ_c^2 the (real) variance of the noise in ct.c. Then, $\sigma_c^2 \leq ct.\sigma^2$ with overwhelming probability. Here, ct.σ designates the variance upper bound stored in the second field of an $\mathcal{E}_H$ ciphertext.*

Remark 1 The unique identifier id of an input can be generated in various ways, depending on whether or not $\mathcal{E}_H$.Enc holds a memory or maintains a state.

Remark 2 Remark that the scheme is *not compact* due to the inclusion of the τ field in the ciphertext format, as the ciphertext size is thus linear in the maximum arity of the possible evaluated functions. However, the decryption function of $\mathcal{E}_H$ does not use that latter field. Therefore, it does not have to be transmitted towards the owner of the secret key to decrypt the results. As such, with respect to the transmission of the (encrypted) end-results, the scheme is compact.

Remark 3 Note that this construction supports multi-hop. The only subtlety in switching to a multi-hop framework is that to evaluate a subsequent circuit, we either need to know the dependencies of the outputs of the previous circuit or to ensure that these outputs, used as inputs for the subsequent circuit, have uncorrelated noises. In the first case, transmitting the τ field in between hops is sufficient. The disadvantage is that the scheme may not be compact in the multi-hop setting. In the second case, we need to ensure that the noises in the output ciphertexts of the first circuit are uncorrelated, then use an Average Case rule at the beginning of a subsequent circuit. To do so, an option is to smudge in between hops. Then, the scheme remains compact but induces a (mild) dependence on $\log_2 q$ on the number of hops. In other words, at the ends, $\log_2 q$ will be $O(H\lambda)$ with H the number of hops.

4.5 CPA^D security of $\mathcal{E}_H$

The following section establishes the CPA^D security of $\mathcal{E}_H$.

Lemma 5 *Given* $\mathsf{ct} = (c, \sigma^2, \tau) \in \mathcal{E}_H.\mathcal{C}$*, when* $\mathcal{E}_H.\mathsf{Dec}(\mathsf{ct}) \neq \perp$*, decryption is correct with probability* $1 - neg(\lambda)$*.*

Proof. After removal of the $a \cdot \mathsf{sk}$ term, the above $\mathcal{E}_H$ decryption algorithm ends up with

$$\Delta m + e + e_{\text{smg}}, \tag{7}$$

leading to a decryption error whenever $||e + e_{\text{smg}}||_\infty \geq \frac{\Delta}{2}$. Since $\sigma_c^2 \leq \mathsf{ct}.\sigma^2$, the smudging lemma (Lemma 3) applies and (7) is indistinguishable from $\Delta m + e_{\text{smg}}$ meaning that, with probability $1 - \text{neg}(\lambda)$, a decryption error may occur iff $||e_{\text{smg}}||_\infty \geq \frac{\Delta}{2}$ (i.e. independently of e). This is precisely the condition under which $\mathcal{E}_H.\mathsf{Dec}(\mathsf{ct})$ returns $\perp$, hence the claim. □

Proposition 1 *Let* $\mathcal{A}$ *be an adversary against the CPA*^{D} *security of* $\mathcal{E}_H$*, then there exists an adversary* $\mathcal{B}$ *against the CPA security of BFV which uses* $\mathcal{A}$ *as a subroutine, and achieves at least the same advantage as* $\mathcal{A}$*.*

Proof. At first $\mathcal{B}$ communicates the public material of the (multi-challenge) CPA challenger[7] to $\mathcal{A}$ as well as initializes an empty internal state $S^{\mathcal{B}} = [\,]$ and sets $\mathsf{ctr} := 0$. Then $\mathcal{B}$ simulates $\mathcal{A}$'s requests as follows:

- When $\mathcal{A}$ issues an *encryption* request with m_0, m_1, $\mathcal{B}$ forwards it to the CPA challenger to get $c = \mathsf{BFV.Enc}(m_b)$. Then, $\mathcal{B}$ updates

$$S^{\mathcal{B}} := \left[S^{\mathcal{B}}; \left(m_0, m_1, (c, \sigma_0^2, \{\mathsf{ctr}\})\right)\right],$$

and sends $(c, \sigma_0^2, \{\mathsf{ctr}\})$ back to $\mathcal{A}$ (and further increments ctr by 1). When $m_0 = m_1$, $\mathcal{B}$ could also process the request on his or her own, using the public-key.

[7] I.e. the multiple challenges variant of (IND-)CPA, also sometimes denoted (LOR-)CPA, or (FTG-)CPA in the terminology of [3, 4] which show the equivalence between the two notions.

- When $\mathcal{A}$ issues an *evaluation* request with function f and game state indices $l_1, ..., l_K$, then $\mathcal{B}$ performs that evaluation step-by-step, staring from $\mathcal{E}_H$-formatted ciphertexts
$$\Big(S^{\mathcal{B}}[l_i].c, S^{\mathcal{B}}[l_i].\sigma^2, S^{\mathcal{B}}[l_i].\tau\Big)$$
following $\mathcal{E}_H$'s rules for the unitary univariate and bivariate operations until ciphertext $(c', {\sigma'}^2, \tau')$ is obtained as an evaluated ciphertext for f. Then, $\mathcal{B}$ performs the left and right cleartext evaluations to get
$$m'_0 = f\Big(S^{\mathcal{B}}[l_1].m_0, ..., S^{\mathcal{B}}[l_K].m_0\Big)$$
and
$$m'_1 = f\Big(S^{\mathcal{B}}[l_1].m_1, ..., S^{\mathcal{B}}[l_K].m_1\Big).$$
Finally, $\mathcal{B}$ updates its internal state by doing
$$S^{\mathcal{B}} := \Big[S^{\mathcal{B}}; (m'_0, m'_1, (c', {\sigma'}^2, \tau'))\Big], \tag{8}$$
and returns $(c', {\sigma'}^2, \tau')$ to $\mathcal{A}$.
- When $\mathcal{B}$ receives a *decryption* request with game state index l, $\mathcal{B}$ checks whether $S^{\mathcal{B}}[l].m_0 = S^{\mathcal{B}}[l].m_1$ and returns $\perp$ when it is not the case. This being done, $\mathcal{B}$ computes $S[l].\sigma = \sqrt{S[l].\sigma^2}$, picks a smudging noise polynomial e_{smg} uniformly in $[\![-2^{\frac{3\lambda}{2}} S[l].\sigma, 2^{\frac{3\lambda}{2}} S[l].\sigma]\!]^n$ and returns $S[l].m_0$ (or equivalently $S[l].m_1$) when $||e_{\text{smg}}||_\infty < \frac{\Delta}{2}$, and $\perp$ otherwise.

The key point in the above, is that $\mathcal{B}$'s handling of $\mathcal{A}$'s decryption requests is indistinguishable from a CPA^D decryption oracle having access to the challenger's secret key. Indeed, by Lemma 5 and the fact that the ciphertext noise variances are correctly bounded[8] by $\mathcal{B}$ (from Lemma 4), whenever $||e_{\text{smg}}||_\infty < \frac{\Delta}{2}$ a decryption error can occur only with probability $1 - \text{neg}(\lambda)$.

Hence, when $\mathcal{B}$ outputs the same guess for b as $\mathcal{A}$, it wins the CPA game with the same advantage as $\mathcal{A}$. □

4.6 Variants of $\mathcal{E}_H$

In this section, we explore additional variants of $\mathcal{E}_H$ which handle in various ways the interactions of ciphertexts with correlated noises during a circuit evaluation. These variants offer different trade-offs between efficiency and reliability.

[8] Recall that, by definition, a CPA^D adversary follows the cryptosystem specification.

4.6.1 Definition of additional variants of $\mathcal{E}_H$

Variant I - $\mathcal{E}_H^{(\text{smg})}$
This first variant adopts the following strategy. When an operation between two ciphertexts with correlated noises is detected, one of the two ciphertexts is smudged before the computation, making the noises of the two ciphertexts (indistinguishable from) independent ones. Thus, the dependencies of the smudged ciphertext are reset to an empty set and its variance is set to that of the smudging noise variance. Only after this step the bivariate operation is performed.

- The $\mathcal{E}_H^{(\text{smg})}$ algorithms for KeyGen, Enc, Eval (univariate op.), Eval ($f \in \mathcal{F}$) and Dec are the same as for $\mathcal{E}_H$.
- $\mathcal{E}_H^{(\text{smg})}.\mathsf{Eval}$ (bivariate op.): given $\omega \in \mathcal{O}_2$ and $\mathsf{ct}_1, \mathsf{ct}_2 \in \mathcal{E}_H^{(\text{smg})}.\mathcal{C}^2$, when $\mathsf{ct}_1.\tau \cap \mathsf{ct}_2.\tau = \emptyset$ return
$$\Big(\mathsf{BFV.Eval}(\omega, \mathsf{ct}_1.c, \mathsf{ct}_2.c), \mathsf{B}_2^{\text{AvgC}}(\omega, \mathsf{ct}_1.\sigma^2, \mathsf{ct}_2.\sigma^2), \mathsf{ct}_1.\tau \cup \mathsf{ct}_2.\tau\Big). \quad (9)$$
Else, pick e_{smg} uniformly from $[\![-2^{\frac{3\lambda}{2}} ct_1.\sigma, 2^{\frac{3\lambda}{2}} ct_1.\sigma]\!]^n$ as a smudging noise polynomial and return
$$\Big(\mathsf{BFV.Eval}(\omega, \mathsf{ct}_1.c + (0, e_{\text{smg}}), \mathsf{ct}_2.c), \mathsf{B}_2^{\text{AvgC}}(\omega, 2^{\frac{3\lambda}{2}} \mathsf{ct}_1.\sigma, \mathsf{ct}_2.\sigma^2), \mathsf{ct}_2.\tau\Big).$$

Variant II - $\mathcal{E}_H^{(\text{dSmg})}$
In this variant, we focus on the decryption process. Here, instead of returning $\perp$ as in $\mathcal{E}_H$ when the smudging noise is too large, the decryption algorithm smudges the ciphertext and returns the decryption output. Note that in this variant, the output of the decryption may be incorrect because of the added smudging noise.

- The $\mathcal{E}_H^{(\text{dSmg})}$ algorithms for KeyGen, Enc, Eval (univariate op.), Eval (bivariate op.) and Eval ($f \in \mathcal{F}$) are the same as for $\mathcal{E}_H$.
- $\mathcal{E}_H^{(\text{dSmg})}.\mathsf{Dec}$: given $\mathsf{ct} = (c, \sigma^2, \tau) \in \mathcal{E}_H.\mathcal{C}$, let $\sigma = \sqrt{\mathsf{ct}.\sigma^2}$ and draw a smudging noise polynomial e_{smg} uniformly from $[\![-2^{\frac{3\lambda}{2}}\sigma, 2^{\frac{3\lambda}{2}}\sigma]\!]^n$. Then, return the result of $\mathsf{BFV.Dec}(\mathsf{ct}.c + (0, e_{\text{smg}}))$.

4.6.2 CPA^D security of these variants

Lemma 6 *Let c_1 and c_2 denote two (R)LWE ciphertexts with correlated noises. Let σ_1^2 and σ_2^2 denote the variances of these latter. Let $\tilde{c}_1$ be obtained from c_1 by adding an independent smudging noise uniformly picked from $[\![-2^{\frac{3\lambda}{2}}\sigma_1, 2^{\frac{3\lambda}{2}}\sigma_1]\!]^n$. Then $\tilde{c}_1$ and c_2 have noises indistinguishable from independent ones.*

Proof. Let e_1 and e_2 be the noises in c_1 and c_2. We denote the smudging noise as $\tilde{e}_1$. Then $\tilde{c}_1$ has noise $e_1 + \tilde{e}_1$. From Lemma 3, ciphertext $\tilde{c}_1 = (a, a \cdot s + \Delta m + e_1 + \tilde{e}_1)$ is indistinguishable from ciphertext $(a, a \cdot s + \Delta m + \tilde{e}_1)$. As a consequence of the independence between $\tilde{e}_1$ and e_2, the pair $\tilde{c}_1$ and c_2 is thus indistinguishable from a pair of ciphertexts with independent noises where the first noise is uniformly picked from $[\![-2^{\frac{3\lambda}{2}}\sigma_1, 2^{\frac{3\lambda}{2}}\sigma_1]\!]^n$ and the second one is of variance σ_2^2, hence the claim. □

Lemma 7 *Let e_{smg} be uniformly picked from $[\![-2^{\frac{3\lambda}{2}}\sigma, 2^{\frac{3\lambda}{2}}\sigma]\!]^n$, and let (a, b) be an RLWE pair with a noise of variance σ^2. Then, $\lceil \frac{1}{\Delta}(\Delta m + e_{\text{smg}}) \rfloor$ is statistically indistinguishable from $\lceil \frac{1}{\Delta}((b + e_{\text{smg}}) - a \cdot \mathsf{sk}) \rfloor$.*

Proof. We have $b + e_{\text{smg}} = a \cdot \mathsf{sk} + \Delta m + e + e_{\text{smg}}$, then

$$\left\lceil \frac{1}{\Delta}((b + e_{\text{smg}}) - a \cdot \mathsf{sk}) \right\rfloor = \left\lceil \frac{1}{\Delta}(\Delta m + e + e_{\text{smg}}) \right\rfloor,$$

is statistically indistinguishable from $\lceil \frac{1}{\Delta}(\Delta m + e_{\text{smg}}) \rfloor$ by Lemma 3. □

The following propositions are corollaries of Proposition 1, and establish the CPA^D security of the variants $\mathcal{E}_H^{(\text{smg})}$ and $\mathcal{E}_H^{(\text{dSmg})}$, under the assumption that BFV is CPA secure.

Corollary 1 (Proposition 1) *BFV CPA security implies $\mathcal{E}_H^{(smg)}$ CPA^D security.*

Proof. Lemma 6 implies that $\mathcal{E}_H^{(\text{smg})}$ satisfies Lemma 4, i.e. that all ciphertext noise variances are correctly bounded. Thus, the CPA^D security of $\mathcal{E}_H^{(\text{smg})}$ follows directly from Proposition 1. □

Corollary 2 (Proposition 1) *BFV CPA security implies $\mathcal{E}_H^{(dSmg)}$ CPA^D security.*

Proof. This is proved with the same reduction as in the proof of Proposition 1, with a modified handling for decryption requests:

- When $\mathcal{B}$ receives a *decryption* request with game state index l, $\mathcal{B}$ checks whether $S^{\mathcal{B}}[l].m_0 = S^{\mathcal{B}}[l].m_1$. The reduction returns $\perp$ when it is not the case. Otherwise, $\mathcal{B}$ computes $S[l].\sigma = \sqrt{S[l].\sigma^2}$, picks a smudging noise polynomial e_{smg} uniformly in $[\![-2^{\frac{3\lambda}{2}}S[l].\sigma, 2^{\frac{3\lambda}{2}}S[l].\sigma]\!]^n$ and returns $\Delta S[l].m_0 + e_{\text{smg}}$ (or equivalently $\Delta S[l].m_1 + e_{\text{smg}}$).

To finalize the proof, Lemma 7 is used instead of Lemma 5 in the indistinguishability argument between $\mathcal{B}$'s handling of $\mathcal{A}$'s decryption request versus $\mathcal{E}_H^{(\text{dSmg})}$ decryption function. □

5 Parameters

Because we achieve CPA^D security by means of guaranteed variance bound calculations and smudging, our variants achieve CPA^D security for any circuits. However, because we do so independently of the ciphertext modulus, these variants may be useless in the sense that decryption errors or $\perp$ answers may occur with large probability. To deal with this we finally instantiate our schemes (i.e. choose the ciphertext modulus q and subsequently n, the latter under the constraint of the target security level) by fixing a noise variance budget σ^2_{max} and select the parameters to achieve a preset small probability of incorrect decryptions. In this context, this latter probability is a *reliability* parameter which is independent of CPA^D security.

Interestingly, for somewhat correct schemes such as BFV, the AAHE paradigm precisely defines the class of allowable functions by a noise variance budget which should not be exceeded. In that sense, with the additional parameters setting methodology defined in the next section, we argue that our approach is an automated instantiation of the AAHE paradigm for somewhat correct FHE schemes (focusing on BFV).

5.1 Reliability of $\mathcal{E}_H$

In this section, we thus look at the parameters for a practical implementation. Recall that, to ensure security, the decryption function picks a smudging noise and outputs $\perp$ when this noise is too large. Therefore, we would like to choose parameters large enough to have a probability of obtaining this non-informative $\perp$ answer around $\text{neg}(k)$ for a specific reliability parameter k (typically $k = 40$). To do so, we consider the case where we want to evaluate algorithms that output ciphertexts of noise variance bounded by σ^2_{max}. To decrypt such ciphertexts, the function $\mathcal{E}_H$.Dec picks a smudging noise e_{smg} uniformly from $[\![-2^{\frac{3\lambda}{2}}\sigma_{\text{max}}, 2^{\frac{3\lambda}{2}}\sigma_{\text{max}}]\!]^n$ and returns $\perp$ if and only if this smudging noise is such that $||e_{\text{smg}}||_\infty \geq \frac{\Delta}{2}$.

The Chebyshev inequality then tells us that $\Pr\left(|X| \geq \frac{\Delta}{2}\right) \leq \left(\frac{2\sigma}{\Delta}\right)^2$, meaning that the probability that *no* RLWE decryption error occurs is[9] such that

$$\left(\Pr\left(|e| < \frac{\Delta}{2}\right)\right)^n \geq \left(1 - \left(\frac{2\sigma}{\Delta}\right)^2\right)^n. \tag{10}$$

As we would like to have a probability to obtain $\perp$ of less than $1 - 2^{-k}$, letting

$$\left(1 - \left(\frac{2\sigma}{\Delta}\right)^2\right)^n = 1 - 2^{-k},$$

yields the following constraint for q,

$$q \geq \frac{2t\text{Var}_{e_{\text{smg}}}}{\sqrt{1 - \sqrt[n]{1 - 2^{-k}}}} = \frac{2t \cdot n \cdot 2^{\frac{3\lambda}{2}}\sigma_{\text{max}}\left(2^{\frac{3\lambda}{2}}\sigma_{\text{max}} + 1\right)}{3\sqrt{1 - \sqrt[n]{1 - 2^{-k}}}}. \tag{11}$$

Still, the above methodology has a slight pitfall as n may have to increase when q increases to maintain the target security level λ. However, because BFV implementations almost always use powers of two for n (and a very small fixed variance σ_0^2 for fresh ciphertexts), it is easy to determine both q and n through trial-and-error. Thus, when we

[9] As this section is on reliability and not security, we suppose here the independence of the noise in the different coefficients. Here, we could also have assumed that the noise followed a Gaussian distribution and used the much tighter Chernoff bound instead of the Chebyshev bound, which would have led to smaller parameters.

have an n and a q for a CPA^D secure but possibly unreliable instantiation of $\mathcal{E}_H$, one may use Eq. (11) to obtain a new larger modulus $\tilde{q}$.

Finally, suppose the set of parameters (n, $\tilde{q}$, σ_0) provides a lower security level than that desired. In that case, one may consider switching to a larger dimension and use ($2n$, $\tilde{q}$, σ_0).

5.2 Reliability of the variants in Sect. 4.6

In this section, we analyze the reliability of the variants of $\mathcal{E}_H$, specifically focusing on how to ensure that the probability of obtaining a non-informative $\perp$ response or an incorrect decryption is below a preset value. As previously, we consider the case where we want to evaluate algorithms that output ciphertexts of noise variance bounded by σ^2_{max}.

To ensure the reliability of the $\mathcal{E}_H^{(\text{smg})}$ variant, we use the same approach as for $\mathcal{E}_H$. Specifically, we use Eq. (11) to ensure a probability of obtaining $\perp$ around neg(k). As for $\mathcal{E}_H^{(\text{dSmg})}$, we want to ensure that the probability of incorrect decryption is bounded by neg(k). Similarly to $\mathcal{E}_H$.Dec, the function $\mathcal{E}_H^{(\text{dSmg})}$.Dec picks a smudging noise e_{smg} uniformly from $\llbracket -2^{\frac{3\lambda}{2}}\sigma_{\text{max}}, 2^{\frac{3\lambda}{2}}\sigma_{\text{max}}\rrbracket^n$, then, it returns $\mathsf{BFV.Dec}(\mathsf{ct}.c + (0, e_{\text{smg}}))$. This decrypts incorrectly when the smudging noise is such that $||e_{\text{smg}}||_\infty \geq \frac{\Delta}{2}$. It is the same condition that ensures the probability of obtaining $\perp$ for $\mathcal{E}_H$. Thus, we use Eq. (11) to choose q, ensuring a probability of obtaining an incorrect decryption less than neg(k).

6 On another three parties setup

Until now, we have assumed a setup in which a (honnest-but-curious) server receives fresh well-formed ciphertexts. This means ciphertexts obtained following genuine executions of the encryption function over genuinely generated keys i.e., following the specifications of the scheme, including the noise distribution.

In this section, we investigate another setup which also represents real-life situations where the client who generates the keys, decrypts and communicates with the server is distinct from the provider who encrypts and provides the input ciphertexts to the homomorphic computations eventually performed by the server. In the next section, we show how to handle this setup such that the data provider is not anymore an adversary against the security of the FHE scheme i.e., in a way such that it does not manipulate or even see any homomorphic ciphertext.

6.1 PRF synchronization to the rescue

Ciphertext noise independence is trivially achieved in the private-key case since ciphertexts are generated with knowledge of the secret key, which then allows to add an independently sampled fresh encryption noise. In this section, we try to benefit from this observation for a three (non-colluding) parties setting where a *server* S (honest-but-curious) provides (encrypted) results of computations performed over encrypted data given by a *provider* P (possibly malicious) to a *client* C (honest-but-curious) with decryption capability. We now show how to achieve ciphertext noise independence for this setup, relying on the private-key setting and avoiding the disclosure of $\mathcal{E}_H.\mathsf{sk}$ to P. To do so, we will use a PRF (Pseudo Random Function) to generate tokens which will be used to mask plaintexts. We will denote by prf_i the i-th term generated by the PRF. The protocol is split into a first offline phase and an online one. The protocol offline phase then goes as follows.

1. Initially, C runs $\mathcal{E}_H.\mathsf{KeyGen}$ to get $\mathcal{E}_H.\mathsf{sk}$ and he also uniformly picks a seed for PRF.
2. Then, C sends seed to P as well as any necessary material for homomorphic evaluation to S.
3. For some constant K and $k \in [\![0, K-1]\!]$, C runs PRF to get prf_k and sends $\mathsf{tk}_k = \mathcal{E}_H.\mathsf{Enc}(\mathsf{prf}_k;\mathsf{sk})$ to S, where tk_k is a triplet composed of one BFV ciphertext $(\mathsf{tk}_k.a, \mathsf{tk}_k.b)$, a noise variance of the BFV fresh ciphertext and a unique identifier. Note that K is assumed to be "large enough". This step may be repeated from time to time when the server runs out of tk_k tokens.

Remark that a second PRF may be used between C and S to avoid sending the $\mathsf{tk}_k.a$'s, thereby reducing the communication burden of step 3 above.

The online phase of the protocol then proceeds as follows:

1. To send a batch of L encrypted inputs $m_k, ..., m_{k+(L-1)}$ to S, P computes $\overline{m_{k+i}} = m_{k+i} - \mathsf{prf}_{k+i} \mod t$ for all $i \in [\![0, L-1]\!]$, and sends the batch of these encrypted inputs.
2. Upon receiving inputs $\overline{m_k}, ..., \overline{m_{k+(L-1)}}$ from P, the server S switches them to the FHE domain in doing[10] $\mathsf{ct}_k = \mathcal{E}_H.\mathsf{AddConst}(\overline{m_k}, \mathsf{tk}_k)$ to get ciphertexts $\mathsf{ct}_k, ...,$ $\mathsf{ct}_{k+(L-1)}$, *all with independent noises* (since the tk_k's have independent noises). With these, S may perform $\mathsf{ct} = \mathcal{E}_H.\mathsf{Eval}(f, \mathsf{ct}_k, ..., \mathsf{ct}_{k+(L-1)})$ for some useful function f, and return ct to C for decryption.

For clarity, both phases are illustrated in Fig. 1 in appendix 14.

Interestingly, in the above protocol, P is not generating or seeing any ciphertext under $\mathcal{E}_H$ and is therefore *not* an adversary against the security of that scheme. Furthermore, even if P and S collude, as long as S operates within the CPA^D model (which is our assumption), they are not able to learn anything they do not already (collectively) know.

In essence, this means that the protocol given in this section is secure in a threat model where P is malicious and S is honest-but-curious. Indeed, P can only provide (masked) plaintexts to the server. Even if P sends an arbitrary value $v \in \mathcal{P}$ to S, as P knows what mask he was supposed to apply (i.e., prf_i), P knows which plaintext he

[10] Recall that the $\mathsf{AddConst}$ operator leaves the ciphertext noise unchanged.

is sending (m is then $v + \mathsf{prf}_i \mod t$), thus achieving an analogue property to that of plaintext awareness.

7 Experimental Results

As a warmup, we first focus in this section on computing linear combinations of L fresh BFV ciphertexts $\mathsf{ct}_0, \ldots, \mathsf{ct}_{L-1}$ using a ternary distribution χ_s for the secret key and the randomness u used for encryption (cf Section 2.3). From Table 1, we have that the variance of the noise e_i of ct_i is $\sigma^2_{\text{fresh}} = \frac{t^2}{q^2}\left(\frac{1}{12} + \sigma_0^2(\frac{4n}{3} + 1)\right)$.

Then, the ciphertext noise variance after applying a linear combination is

$$\begin{aligned}\sigma^2_{\text{max}} &:= \text{Var}\left[\sum_{i=0}^{L-1} \alpha_i e_i\right] = \sum_{i=0}^{L-1} \alpha_i^2 \sigma^2_{\text{fresh}} \\ &= \frac{t^2}{q^2}\left(\frac{1}{12} + \sigma_0^2\left(\frac{4n}{3} + 1\right)\right)\sum_{i=0}^{L-1} \alpha_i^2\end{aligned}$$

We refer to $||\alpha||^2 = \sum_{i=0}^{L-1} \alpha_i^2$ as the L_2-*budget*. Following the above calculation, one may perform either $||\alpha||^2$ additions of ciphertexts with *independent* noises or only $||\alpha||$ additions of a given ciphertext with itself, and then achieve 2^{-k} probability of getting $\perp$ upon decryption. Table 2 provides some examples of parameters for $\mathcal{E}_H$, using Eq. (11) for choosing the parameter q. As an illustration, a ciphertext modulus q on around 112 bits allows to sum up to 1000 independent ciphertexts or up to around 30 times the given ciphertext with itself. In both cases, there is a probability of less than 2^{-40} of getting $\perp$ at decryption of the result.

Also, in Table 2, the left-hand side represents the L_2-budget for the sum of 1000 independent ciphertexts, while the right-hand side represents the L_2-budget for the sum of 10^6 independent ciphertexts, thus for $\sqrt{10^6} = 1000$ additions of a given ciphertext with itself. Consistently, we observe that the parameters to choose are larger when we have to add a ciphertext to itself 1000 times, compared to performing the sum of 1000 independent ciphertexts. The runtime to evaluate this sum is of 24.035 ms using the closest by default OpenFHE parameters, i.e. $n = 8192$, $t = 2^{16} + 1$ and $\log_2 q = 120$.

t	$\log_2 q$	n	λ	t	$\log_2 q$	n	λ
2	97	8192	284	2	99	8192	277
256	104	8192	260	256	106	8192	254
2^{16}	112	8192	236	2^{16}	114	8192	231
2^{32}	128	8192	198	2^{32}	130	8192	194
2^{32}	129	16384	480	2^{32}	131	16384	471
2^{64}	161	16384	364	2^{64}	163	16384	359

Table 2 Example of parameters for $\mathcal{E}_H$, in function of the plaintext modulus t. For a L_2-*budget* of 10^3 (left half) and 10^6 (right half), and $\sigma_0 = 3.19$, $k = 40$. Estimated security levels have been computed by means of the lattice-estimator.

Table 3 provides more extensive results for different parameters sets related to the evaluation of several functions, chosen from the same corpus of functions used in [26]: the computation of the mean and the variance, as well as the logistic and exponential functions using their Maclaurin series of degree 5. For more details, for the logistic regression we computed $\frac{1}{2} + \frac{x}{4} - \frac{x^3}{48} + \frac{x^5}{480}$ and for the exponentiation we computed $1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120}$. To obtain these results, we built a software layer on top of OpenFHE which allows to perform basic operations while tracking the dependency of the noises of the involved ciphertexts and calculating each time the resulting variance bound (with the formulas for worst and average case as described in Section 4).

Function	**Independent inputs**					**Dependent inputs**				
	t	$\log_2 q$	n	λ	time (ms)	t	$\log_2 q$	n	λ	time (ms)
Mean of 100 values	65537	240	16384	217	34	65537	360	16384	124	50
Mean of 500 values	65537	240	16384	217	190	65537	720	32768	124	1450
Variance of 100 values	65537	300	16384	161	3430	65537	360	16384	124	4380
Variance of 500 values	65537	300	16384	217	31650	65537	780	32768	111	135450
Logistic Regression deg 5	65537	360	16384	124	175	65537	360	16384	124	175
Exponential deg 5	65537	360	16384	124	175	65537	360	16384	124	175

Table 3 Results of the homomorphically evaluation of different functions with our approach on top of OpenFHE. Estimated security levels have been computed by means of the lattice-estimator.

As shown in Table 3, we can see that, when performing an average case noise estimation strategy (for the case where all input ciphertexts are independent), the parameters are larger than the ones for a baseline OpenFHE implementation (inducing, for example, an increase in execution time by a factor of 1.3 and 1.5 for the exponential and logistic regression, respectively), in order to absorb the appropriate smudging, but do not change significantly from one circuit to another. Indeed, for the variance calculations, the parameters are a bit larger (larger ciphertext modulus q) than for the average computations, since the inputs values have to be squared, but except this, the parameters do not differ much from one another. To compute the logistic regression and the exponential function of degree 5, an even larger ciphertext modulus q is needed, but this is quite reasonable, since these circuits have a larger multiplicative depth in order to calculate the Maclaurin series up to degree 5.

We can see the additional cost when the inputs are no longer independent and worst-case formulas are used. This can be seen in particular in the first four rows of table 3: when the number of additions is large, the variance increases significantly, since each addition multiplies it by a factor 4. Also, to absorb the smudging appropriate for this case, we need to increase the parameters even further. But then, as we can see on the rows representing the computation of the mean or variance of 500 values, when q needs to be much larger, the parameter n also needs to be increased (then OpenFHE may even require the plaintext modulus t to be increased as well), leading to a much longer execution time for a simple linear combination.

Finally, we can see that, for the logistic regression and the exponential function calculations using the Maclaurin series, the parameters and runtimes for the average-case and the worst-case noise estimation strategies are identical. This is consistent, since the parameter q is already larger for the average-case due to the multiplicative depth, and the circuit requires only 4 multiplications and at most 5 additions (and multiplications by constant), which does not cause a drastic increase in the smudging noise variance.

8 Threshold variants?

Our BFV variants can be naturally adapted to a threshold setting, where each user has a public/private-key pair. During an offline setup, users collaborate to create a joint public-key, paired to a joint private-key split among them. Data is encrypted with the joint public-key and computations are evaluated under a joint evaluation key. To decrypt a message, a minimum number of users, the *threshold*, must collaborate and combine their *partial decryptions*. These are computed by each user then broadcasted to the others.

Note that, in a threshold setting, the use of the smudging technique is mandatory during the partial decryption process, to ensure the security of the user's secret key. Thus, as soon as an appropriate smudging is done during the partial decryption process, i.e. a smudging noise depending on both the ciphertext noise variance and the security parameter, this multi-party setting is CPA^D secure. In our variants, both decryption functions embed a smudging process. As a last remark, note that the $\mathcal{E}_H^{(\mathrm{dSmg})}.\mathsf{Dec}$ seen as a partial decryption merely smudges the ciphertext before decrypting it, thus depicting precisely the decryption algorithm (Collaborative Key Switch) described by Mouchet et al. in [29].

9 Conclusion and future work

In this paper we presented a complete framework for achieving CPA^D security for BFV proposing an approach based on a close monitoring of the noise variance and on the smudging technique. Our approach extends previous works on the AAHE paradigm and provides a methodology for an automatic setting of parameters ensuring a preset reliability target for decryption.

We also provided concrete parameters and execution timings (over OpenFHE) for our main BFV variant in the case of the computation of a linear combination over fresh ciphertexts and other more complex functions, taking into account a reliability parameter and showing the importance of considering the ciphertexts dependencies.

Complementary, we investigated a more advanced setting, involving three non colluding parties: a honest-but-curious server computing homomorphic results for a client on encrypted data coming from some possibly malicious provider. We examined it in a private-key setting, for which we proposed a protocol using a PRF to prevent control of the LWE encryption noise from the malicious provider.

In the future, we envision to integrate this framework in existing homomorphic libraries like OpenFHE or Lattigo in order to relieve the user from the burden of achieving CPA^D security by hand, including in the threshold setting.

Acknowledgements

The authors would like thank Damien Stehlé and Adeline Roux-Langlois for providing feedback on a earlier version of this work (as included in [14]).

References

1. Alexandru, A., Badawi, A.A., Micciancio, D., Polyakov, Y.: Application-Aware Approximate Homomorphic Encryption: Configuring FHE for Practical Use. Tech. Rep. 2024/203, IACR ePrint (2024)
2. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE. In: EUROCRYPT, pp. 483–501 (2012)
3. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: IEEE SFCS, pp. 394–403 (1997)
4. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: CRYPTO, pp. 26–45 (1998)
5. Bendlin, R., Damgård, I.: Threshold Decryption and Zero-Knowledge Proofs for Lattice-Based Cryptosystems. In: TCC, pp. 201–218 (2010)
6. Bergamaschi, F., Costache, A., Dachman-Soled, D., Kippen, H., LaBuff, L., Tang, R.: Revisiting the Security of Approximate FHE with Noise-Flooding Countermeasures. Tech. Rep. 2024/424, IACR ePrint (2024)
7. Bernard, O., Joye, M., Smart, N.P., Walter, M.: Drifting Towards Better Error Probabilities in Fully Homomorphic Encryption Schemes. Tech. Rep. 2024/1718, IACR ePrint (2024)
8. Biasioli, B., Marcolla, C., Calderini, M., Mono, J.: Improving and Automating BFV Parameters Selection: An Average-Case Approach. Tech. Rep. 2023/600, IACR ePrint (2023)
9. Boneh, D., Gennaro, R., Goldfeder, S., Jain, A., Kim, S., Rasmussen, P.M.R., Sahai, A.: Threshold Cryptosystems from Threshold Fully Homomorphic Encryption. In: CRYPTO, p. 565–596 (2018)
10. Boudgoust, K., Scholl, P.: Simple Threshold (Fully Homomorphic) Encryption from LWE with Polynomial Modulus. In: ASIACRYPT (2023)
11. Brakerski, Z.: Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In: CRYPTO, pp. 868–886 (2012)
12. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) Fully Homomorphic Encryption without Bootstrapping. TOCT pp. 1–36 (2014)
13. Brzuska, C., Canard, S., Fontaine, C., Phan, D.H., Pointcheval, D., Renard, M., Sirdey, R.: Relations among new CCA security notions for approximate FHE. IACR Commun. Cryptol. (2025)
14. Checri, M.: Contributions to fhe security in the CPA^D model. Ph.D. thesis, Université Paris-Saclay (2025)
15. Checri, M., Sirdey, R., Boudguiga, A., Bultel, J.P.: On the Practical CPAD Security of "Exact" and Threshold FHE Schemes. In: CRYPTO, pp. 3–33 (2024)
16. Cheon, J.H., Choe, H., Passelègue, A., Stehlé, D., Suvanto, E.: Attacks Against the IND-CPA$^{\mathrm{D}}$ Security of Exact FHE Schemes. In: CCS, pp. 2505–2519 (2024)
17. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic Encryption for Arithmetic of Approximate Numbers. In: ASIACRYPT, pp. 409–437
18. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster Fully Homomorphic Encryption: bootstrapping in Less Than 0.1 Seconds. In: ASIACRYPT (2016)
19. Chowdhury, S., Sinha, S., Singh, A., Mishra, S., Chaudhary, C., Patranabis, S., Mukherjee, P., Chatterjee, A., Mukhopadhyay, D.: Efficient Threshold FHE for Privacy-Preserving Applications. Tech. Rep. 2022/165, IACR ePrint (2022)
20. Costache, A., Laine, K., Player, R.: Evaluating the Effectiveness of Heuristic Worst-Case Noise Analysis in FHE. In: ESORICS, p. 546–565 (2020)

21. Costache, A., Smart, N., Sako, K.: Which ring based somewhat homomorphic encryption scheme is best? (2016-01-01)
22. Fan, J., Vercauteren, F.: Somewhat Practical Fully Homomorphic Encryption. Tech. Rep. 2012/144, IACR ePrint (2012)
23. Guo, Q., Nabokov, D., Suvanto, E., Johansson, T.: Key recovery attacks on approximate homomorphic encryption with nonworst-case noise flooding countermeasures. In: Usenix Security (2024)
24. Iliashenko, I., Preneel, B., Vercauteren, F.: Optimisations of Fully Homomorphic Encryption (2019-05-27)
25. Kim, A., Polyakov, Y., Zucca, V.: Revisiting Homomorphic Encryption Schemes for Finite Fields. p. 608–639. Springer-Verlag (2021)
26. Li, B., Miccianccio, D.: On the Security of Homomorphic Encryption on Approximate Numbers. In: EUROCRYPT, pp. 648–677 (2021)
27. Li, B., Micciancio, D., Schultz, M., Sorrell, J.: Securing Approximate Homomorphic Encryption Using Differential Privacy. In: CRYPTO, p. 560–589 (2022)
28. Micciancio, D., Suhl, A.: Simulation-Secure Threshold PKE from LWE with Polynomial Modulus. IACR Commun. Cryptol. (2024)
29. Mouchet, C., Troncoso-Pastoriza, J., Bossuat, J.P., Hubaux, J.P.: Multiparty Homomorphic Encryption from Ring-Learning-with-Errors. In: PoPETS, pp. 291–311 (2021)
30. Passelègue, A., Stehlé, D.: Low communication threshold fully homomorphic encryption. In: ASIACRYPT, pp. 297–329 (2024)

10 CPA and CCA1 Security Games

In the *CPA game*, the adversary only has access to encryption requests and can perform a unique challenge request. Note that the encryption oracle is necessary only in the private-key case as, in the public-key case, the adversary can generate ciphertexts on its own. In the CPA game, there is also no need for an evaluation oracle since the adversary can always perform homomorphic evaluation on its own and there is no need to fill a game state (recall that in the CPA^D game, the purpose of the encryption and evaluation oracle is to fill the game state with well-formed ciphertexts for handling subsequent decryption request on state indices).

IND-CPA game

Setup: Challenger generates a key $\mathsf{sk} \leftarrow \mathsf{KeyGen}()$.
Phase 1 (Before Challenge): Adversary $\mathcal{A}$ may query an encryption oracle $\mathcal{O}^{\mathsf{Enc}}$ for any message m of its choice.
Challenge: Adversary $\mathcal{A}$ submits two distinct messages m_0, m_1 of equal length.
Challenger selects a random bit $b \leftarrow \{0, 1\}$ and computes $c^* = \mathsf{Enc}_{\mathsf{sk}}(m_b)$.
Challenger sends c^* to $\mathcal{A}$.
Phase 2 (After Challenge): Adversary $\mathcal{A}$ may continue to query the encryption oracle.
Guess: Adversary outputs a guess $b' \in \{0, 1\}$.
Winning Condition: $\mathcal{A}$ wins if $b' = b$, meaning it successfully distinguished whether c^* was an encryption of m_0 or m_1.

In the *CCA1 game*, the adversary has access to encryption requests and can also perform a single challenge request. *Before* this unique challenge request, the adversary is additionally granted access to a first step decryption oracle which simply proceeds as follows: when $\mathcal{A}$ queries $(\texttt{ciphertext}, c)$, it returns her $\mathsf{Dec}(c)$.

Then, *after* the single challenge request, the decryption oracle systematically replies $\perp$. Note that the CCA1 game has no evaluation oracle as the adversary performs the homomorphic evaluations on its own in both the private and public-key setting and there is no need to fill a game state (since the decryption oracle accepts arbitrary ciphertexts rather than indices pointing to well-formed ciphertexts stored in a game state).

IND-CCA1 game

Setup: Challenger generates a key $\mathsf{sk} \leftarrow \mathsf{KeyGen}()$.
Phase 1 (Before Challenge): Adversary $\mathcal{A}$ may query an encryption oracle $\mathcal{O}^{\mathsf{Enc}}$ for any message m of its choice. $\mathcal{A}$ also may query a decryption oracle $\mathcal{O}^{\mathsf{Dec}}$ for any ciphertext c of its choice.
Challenge: Adversary $\mathcal{A}$ submits two distinct messages m_0, m_1 of equal length.
Challenger selects a random bit $b \leftarrow \{0,1\}$ and computes $c^* = \mathsf{Enc}_{\mathsf{sk}}(m_b)$.
Challenger sends c^* to $\mathcal{A}$.
Phase 2 (After Challenge): Adversary $\mathcal{A}$ may continue to query the encryption oracle, but the decryption oracle is now closed.
Guess: Adversary outputs a guess $b' \in \{0,1\}$.
Winning Condition: $\mathcal{A}$ wins if $b' = b$, meaning it successfully distinguished whether c^* was an encryption of m_0 or m_1.

11 BFV homomorphic operators

Recall section 2.3, the BFV scheme homomorphic operators are then defined as follows.

- $\mathsf{BFV.Add}(\mathbf{c} = (c_0, c_1), \mathbf{c}' = (c_0', c_1'))$: compute and return $\mathbf{c}_{\text{add}} = \left([c_0 + c_0']_q, [c_1 + c_1']_q\right)$
- $\mathsf{BFV.MultConst}(\alpha, \mathbf{c} = (c_0, c_1))$: compute and return $\mathbf{c}_{\text{mulc}} = \left([\alpha c_0]_q, [\alpha c_1]_q\right)$
- $\mathsf{BFV.Mult}(\mathbf{c} = (c_0, c_1), \mathbf{c}' = (c_0', c_1'))$: compute and return $\mathbf{c}_{\text{mul}} = \left(\left[\left\lfloor \frac{t}{q} c_0 c_0' \right\rceil\right]_q, \left[\left\lfloor \frac{t}{q} (c_0 c_1' + c_1 c_0') \right\rceil\right]_q, \left[\left\lfloor \frac{t}{q} c_1 c_1' \right\rceil\right]_q\right)$
- $\mathsf{BFV.RelinKeyGen}^{\text{GHS}}(s, p)$: sample $a \overset{\$}{\leftarrow} R_q$, $e \overset{\$}{\leftarrow} \chi$. Set the relinearization key $\mathsf{rlk} = \left(a, [-a \cdot s + e + p \cdot s^2]_{p \cdot q}\right)$.
- $\mathsf{BFV.Relin}^{\text{GHS}}(\mathsf{rlk} = (\mathsf{rlk}.a, \mathsf{rlk}.b), p, \mathbf{c} = (c_0, c_1, c_2))$: compute and return $\mathbf{c}_{\text{relin}} = \left(\left[c_0 + \left\lfloor \frac{c_2 \mathsf{rlk}.a}{p} \right\rceil\right]_q, \left[c_1 + \left\lfloor \frac{c_2 \mathsf{rlk}.b}{p} \right\rceil\right]_q\right)$

12 Noises covariance in and between BFV ciphertexts

In this section, we first check that the covariance between the noises in the same slot of two distinct fresh well-formed BFV ciphertexts is zero (despite of their dependency to the public-key material). This essentially establishes that these noises are uncorrelated and means that we can safely apply the average case variance monitoring strategy (as $V[X+Y] = V[X] + V[Y]$ is also true for uncorrelated random variables) when two distinct fresh well-formed BFV ciphertexts go through some homomorphic operations. We do so for both the conventionnal BFV public-key as well as for the variant with a Regev-style public-key, even if we do not use the latter in this work. In both cases, we also study the covariance between noises in distinct slots of the same well-formed ciphertext. This time the covariance is not necessarily 0.

In the following subsections, unless otherwise specified, we assume that u's are picked at random in $\{0,1\}^n$.

12.1 Noise covariance calculation for BFV (Regev-style public-key).

We focus on coefficient i. Let N denote the size of the public-key and $e_i^{(0)}, ..., e_i^{(N-1)}$ the noises for that coefficient in the RLWE pairs forming the public-key. Then if we consider two ciphertexts, we have noises $e_i = \sum_{k=0}^{N-1} u_k e_i^{(k)}$ and $e'_i = \sum_{k=0}^{N-1} u'_k e_i^{(k)}$. So let us compute Cov$(e_i, e'_i) = E[e_i e'_i] - E[e_i]E[e'_i]$. We have

$$E[e_i] = \sum_{k=0}^{N-1} e_i^{(k)} E[u_k] = \frac{1}{2} \sum_{k=0}^{N-1} e_i^{(k)} = E[e'_i].$$

Then, for u_k uniformly picked in $\{0,1\}$

$$\begin{aligned} E[e_i e'_i] &= \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} e_i^{(k)} e_i^{(l)} E[u_k u'_l] \\ &= \frac{1}{4} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} e_i^{(k)} e_i^{(l)} \end{aligned}$$

and finally

$$\text{Cov}(e, e') = \frac{1}{4} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} e_i^{(k)} e_i^{(l)} - \left(\frac{1}{2} \sum_{k=0}^{N-1} e_i^{(k)} \right)^2 = 0.$$

Now, we consider the covariance between coefficients $i \neq j$ in the *same* fresh ciphertext. In this case we have $e_i = \sum_k u_k e_i^{(k)}$ and $e_j = \sum_k u_k e_j^{(k)}$. For u_k uniformly picked in $\{0,1\}$, we have
$E[e_i] = \sum_k e_i^{(k)} E[u_k] = \frac{1}{2} \sum_k e_i^{(k)}$, and $E[e_j] = \frac{1}{2} \sum_k e_j^{(k)}$.

Then,

$$\begin{aligned}
E[e_i e_j] &= \sum_k \sum_l e_i^{(k)} e_j^{(l)} E[u_k u_l], \\
&= \sum_k e_i^{(k)} e_j^{(k)} E[u_k^2] + \sum_k \sum_{l \neq k} e_i^{(k)} e_j^{(l)} E[u_k u_l], \\
&= \frac{1}{2} \sum_k e_i^{(k)} e_j^{(k)} + \frac{1}{4} \sum_k \sum_{l \neq k} e_i^{(k)} e_j^{(l)}.
\end{aligned}$$

So we end up with,

$$\begin{aligned}
\text{Cov}(e_i, e_j) &= \frac{1}{2} \sum_k e_i^{(k)} e_j^{(k)} + \frac{1}{4} \sum_k \sum_{l \neq k} e_i^{(k)} e_j^{(l)} \\
&\quad - \left(\frac{1}{2} \sum_k e_i^{(k)} \right) \left(\frac{1}{2} \sum_l e_j^{(l)} \right), \\
&= \frac{1}{4} \sum_k e_i^{(k)} e_j^{(k)},
\end{aligned}$$

which is unlikely to be 0.

12.1.1 Noise covariance calculation for BFV (standard public key)

We consider a BFV encryption of 0 i.e.

$$(\underbrace{a_0 \cdot u + e_1}_{a}, \underbrace{b_0 \cdot u + e_2}_{b}),$$

where (a_0, b_0) (with $b_0 = a_0 \cdot s + e_0)$) is the public key, where e_1 and e_2 have their coefficients picked following χ_{σ_0}. Then the noise polynomial ε in that ciphertext is

$$\begin{aligned}
b - a \cdot s &= b_0 \cdot u + e_2 - (a_0 \cdot u + e_1) \cdot s \\
&= b_0 \cdot u + e_2 - a_0 \cdot u \cdot s - e_1 \cdot s \\
&= (\underbrace{b_0 - a_0 \cdot s}_{e_0}) \cdot u + e_2 - e_1 \cdot s \\
&= e_0 \cdot u + e_2 - e_1 \cdot s.
\end{aligned}$$

Now, the i-th coefficient of ε is

$$\begin{aligned}\varepsilon_i = &\sum_{k=0}^{i} e_{0,i-k}u_k - \sum_{k=i+1}^{n-1} e_{0,i+n-k}u_k \\ &- \sum_{k=0}^{i} e_{1,i-k}s_k + \sum_{k=i+1}^{n-1} e_{1,i+n-k}s_k \\ &+ e_{2,i}.\end{aligned} \tag{12}$$

To simplify things, we now have a look at

$$\varepsilon_{n-1} = \sum_{k=0}^{n-1} e_{0,n-1-k}u_k - \sum_{k=0}^{n-1} e_{1,n-1-k}s_k + e_{2,n-1}.$$

Remark that,

$$\begin{aligned}E[\varepsilon_{n-1}] &= \sum_{k=0}^{n-1} e_{0,n-1-k}E[u_k] - \sum_{k=0}^{n-1} E[e_{1,n-1-k}]s_k + E[e_{2,n-1}] \\ &= \sum_{k=0}^{n-1} e_{0,n-1-k}E[u_k]. \end{aligned} \tag{13}$$

Given two fresh ciphertexts (a, b) and $(a'b')$, we now wish to compute

$$\mathrm{Cov}(\varepsilon_{n-1}, \varepsilon'_{n-1}) = E[\varepsilon_{n-1}\varepsilon'_{n-1}] - E[\varepsilon_{n-1}]E[\varepsilon'_{n-1}]. \tag{14}$$

Recall that for two independent random variables X and Y, Cov$(X, Y) = 0$, hence $E[XY] = E[X]E[Y]$. Hence, if one of these two variables is furthermore centered, $E[XY] = 0$. In the present case, u, e_1 and e_2 are independent, e_1 and e_2 are centered and all other involved value are fixed. So,

$$\begin{aligned}E[\varepsilon_{n-1}\varepsilon'_{n-1}] &= E\Bigg[\Bigg(\sum_{k=0}^{n-1} e_{0,n-1-k}u_k - \sum_{k=0}^{n-1} e_{1,n-1-k}s_k + e_{2,n-1}\Bigg) \\ &\qquad \Bigg(\sum_{l=0}^{n-1} e_{0,n-1-l}u'_l - \sum_{l=0}^{n-1} e'_{1,n-1-l}s_l + e'_{2,n-1}\Bigg)\Bigg] \\ &= \sum_{k=0}^{n-1}\sum_{l=0}^{n-1} e_{0,n-1-k}e_{0,n-1-l}E[u_k u'_l] \\ &\quad - \sum_{k=0}^{n-1}\sum_{l=0}^{n-1} s_k e_{0,n-1-l}\underbrace{E[e_{1,n-1-k}u'_l]}_{=0} \\ &\quad + \sum_{k=0}^{n-1} e_{0,n-1-k}\underbrace{E[e'_{2,n-1}u_k]}_{=0}\end{aligned}$$

$$
\begin{aligned}
&- \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} s_k e_{0,n-1-l} \underbrace{E[e_{1,n-1-k} u'_l]}_{=0} \\
&+ \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} s_k s_l \underbrace{E[e_{1,n-1-k} e'_{1,n-1-l}]}_{=0} \\
&- \sum_{k=0}^{n-1} s_k \underbrace{E[e'_{2,n-1} e_{1,n-1-k}]}_{=0} \\
&+ \sum_{l=0}^{n-1} e_{0,n-1-l} \underbrace{E[e_{2,n-1} u'_l]}_{=0} \\
&- \sum_{l=0}^{n-1} s_l \underbrace{E[e'_{1,n-1-l} e_{2,n-1}]}_{=0} \\
&+ \underbrace{E[e_{2,n-1} e'_{2,n-1}]}_{=0} \\
= &\sum_{k=0}^{n-1} \sum_{l=0}^{n-1} e_{0,n-1-k} e_{0,n-1-l} E[u_k u'_l]
\end{aligned}
$$

It then follows from Eq. (13) and (14) that,

$$
\begin{aligned}
\mathrm{Cov}(\varepsilon_{n-1}, \varepsilon'_{n-1}) = &\sum_{k=0}^{n-1} \sum_{l=0}^{n-1} e_{0,n-1-k} e_{0,n-1-l} E[u_k u'_l] \\
&- \left(\sum_{k=0}^{n-1} e_{0,n-1-k} E[u_k] \right)^2
\end{aligned}
$$

For u_k, u'_l uniformly picked in $\{0;1\}^n$

$$
\begin{aligned}
\mathrm{Cov}(\varepsilon_{n-1}, \varepsilon'_{n-1}) &= \frac{1}{4} \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} e_{0,n-1-k} e_{0,n-1-l} \\
&\quad - \left(\frac{1}{2} \sum_{k=0}^{n-1} e_{0,n-1-k} \right)^2 \\
&= 0.
\end{aligned}
$$

For u_k, u'_l uniformly picked in $\{-1, 0; 1\}^n$

$$\begin{aligned}
\mathrm{Cov}(\varepsilon_{n-1}, \varepsilon'_{n-1}) &= \sum_{k=0}^{n-1}\sum_{l=0}^{n-1} e_{0,n-1-k}e_{0,n-1-l}\cdot 0 \\
&\quad - \Bigg(\sum_{k=0}^{n-1} e_{0,n-1-k}\cdot 0\Bigg)^2 \\
&= 0.
\end{aligned}$$

We now would like to investigate $\mathrm{Cov}(\varepsilon_i, \varepsilon_j)$ $(i \neq j)$ i.e., the noise covariance between different slots in the same ciphertext. To simplify things, we look here at $\mathrm{Cov}(\varepsilon_{n-1}, \varepsilon_0)$. Note that similar computations can be done for $\mathrm{Cov}(\varepsilon_i, \varepsilon_j)$, but for readability, we focus on the $n-1$ and 0 coefficients, because the calculations are lighter. For the last part of this computation, we suppose that u is uniformly picked[11] in $\{0,1\}^n$. Following Eq. (12), we have

$$\varepsilon_0 = e_{0,0}u_0 - \sum_{k=1}^{n-1} e_{0,n-k}u_k - e_{1,0}s_0 + \sum_{k=1}^{n-1} e_{1,n-k}s_k + e_{2,0},$$

and, thus,

$$E[\varepsilon_0] = \frac{1}{2}\Bigg(e_{0,0} - \sum_{k=1}^{n-1} e_{0,n-k}\Bigg).$$

Now,

$$\begin{aligned}
E[\varepsilon_{n-1}\varepsilon_0] &= E\Bigg[\Bigg(\sum_{k=0}^{n-1} e_{0,n-1-k}u_k - \sum_{k=0}^{n-1} e_{1,n-1-k}s_k + e_{2,n-1}\Bigg) \\
&\qquad \Bigg(e_{0,0}u_0 - \sum_{l=1}^{n-1} e_{0,n-l}u_l - e_{1,0}s_0 \\
&\qquad + \sum_{l=1}^{n-1} e_{1,n-l}s_l + e_{2,0}\Bigg)\Bigg] \\
&= \sum_{k=0}^{n-1} e_{0,0}e_{0,n-1-k}E[u_0u_k] \\
&\quad - \sum_{k=0}^{n-1}\sum_{l=1}^{n-1} e_{0,n-1-k}e_{0,n-l}E[u_ku_l] \\
&\quad + \sum_{k=0}^{n-1} s_0s_kE[e_{1,0}e_{1,n-1-k}] \\
&\quad - \sum_{k=0}^{n-1}\sum_{l=1}^{n-1} s_ls_kE[e_{1,n-1-k}e_{1,n-l}]
\end{aligned}$$

[11] In this section, we chose a binary distribution for u, but the same reasoning applies to a ternary distribution, with u uniformly picked in $\{-1,0,1\}^n$

$$
\begin{aligned}
&= e_{0,0}e_{0,n-1}E[u_0^2] + \sum_{k=1}^{n-1} e_{0,0}e_{0,n-1-k}E[u_0u_k] \\
&\quad - \sum_{k=1}^{n-1} e_{0,n-1-k}e_{0,n-k}E[u_k^2] \\
&\quad - \sum_{k=0}^{n-1} \sum_{l=1,l\neq k}^{n-1} e_{0,n-1-k}e_{0,n-l}E[u_ku_l] \\
&\quad + s_0s_{n-1}E[e_{1,0}^2] \\
&\quad - \sum_{k=1}^{n-1} s_ks_{k-1}E[e_{1,n-k}^2], \\
&= \frac{1}{2}\left(e_{0,0}e_{0,n-1} - \sum_{k=1}^{n-1} e_{0,n-1-k}e_{0,n-k}\right) \\
&\quad + \frac{1}{4}\left(\sum_{k=1}^{n-1} e_{0,0}e_{0,n-1-k} - \sum_{k=0}^{n-1}\sum_{l=1,l\neq k}^{n-1} e_{0,n-1-k}e_{0,n-l}\right) \\
&\quad + \sigma_0^2\left(s_0s_{n-1} - \sum_{k=1}^{n-1} s_ks_{k-1}\right).
\end{aligned}
$$

Then,

$$
\begin{aligned}
E[\varepsilon_{n-1}]E[\varepsilon_0] &= \left(\frac{1}{2}\sum_{k=0}^{n-1} e_{0,n-1-k}\right)\left(\frac{1}{2}\left(e_{0,0} - \sum_{l=1}^{n-1} e_{0,n-l}\right)\right), \\
&= \frac{1}{4}\left(e_{0,0}\sum_{k=0}^{n-1} e_{0,n-1-k} - \sum_{k=0}^{n-1}\sum_{l=1}^{n-1} e_{0,n-1-k}e_{0,n-l}\right)
\end{aligned}
$$

Thus,

$$
\begin{aligned}
\text{Cov}(\varepsilon_{n-1}, \varepsilon_0) &= \frac{1}{4}\left(e_{0,0}e_{0,n-1} - \sum_{k=1}^{n-1} e_{0,n-1-k}e_{0,n-k}\right) \\
&\quad + \sigma_0^2\left(s_0s_{n-1} - \sum_{k=1}^{n-1} s_ks_{k-1}\right),
\end{aligned}
$$

a quantity that is also unlikely to be exactly 0.

13 Noise variance analysis

13.1 Noise variance induced by (public-key) BFV encryption

We first recall how the public key is generated. χ_e is the distribution from which the coefficients of noise polynomials are picked and has variance v_e. χ_s is the distribution used to pick the coefficients of the secret key polynomial and has variance v_s.

$$pk = (b = -a \cdots + e, a)$$

with $a \overset{\$}{\leftarrow} \mathcal{R}_q$ and $e \leftarrow \chi_e$. Encrypting a message m using the public-key is done as follows:

$$\mathsf{Enc}(m) = c = (c_0, c_1) = \left(b \cdot u + \left\lceil \frac{q}{t} m \right\rfloor + e_0, a \cdot u + e_1\right)$$

with $u \leftarrow \chi_s$ and $e_0, e_1 \leftarrow \chi_e$.

The phase of the resulting ciphertext is:

$$\begin{aligned}
\phi(c) &= c_0 + c_1 s \\
&= b \cdot u + \left\lceil \frac{q}{t} m \right\rfloor + e_0 + (a \cdot u + e_1) \cdot s \\
&\text{with } \epsilon \text{ being a polynomial with cofficients following } U_{[-\frac{1}{2}, \frac{1}{2}]} \\
&= (-a \cdot s + e) \cdot u + \frac{q}{t} m + \epsilon + e_0 + a \cdot u \cdot s + e_1 \cdot s \\
&= \frac{q}{t} m + e \cdot u + \epsilon + e_0 + e_1 \cdot s
\end{aligned}$$

The noise in this fresh ciphertext is thus $e \cdot u + \epsilon + e_0 + e_1 \cdot s$. We now analyze the variance of the coefficients of this noise. Let $i \in [\![0; n-1]\!]$,

$$\begin{aligned}
&\mathrm{Var}\left[\left(e \cdot u + \epsilon + e_0 + e_1 \cdot s\right)_{|i}\right] \\
&= \mathrm{Var}\left[\left(e \cdot u\right)_{|i}\right] + \mathrm{Var}[\epsilon_{|i}] + \mathrm{Var}[e_{0\ |i}] + \mathrm{Var}\left[\left(e_1 \cdot s\right)_{|i}\right] \\
&= n v_e v_s + \frac{1}{12} + v_e + n v_e v_s \\
&= \frac{1}{12} + v_e(2 n v_s + 1)
\end{aligned}$$

Remark 4 When multiplying two polynomials a and b modulo $X^n + 1$, the i-th coefficient of the resulting polynomial, denoted $(ab)_{|i}$ is computed as follows: $(ab)_{|i} = \sum_{k=0}^{i} a_k \cdot b_{i-k} - \sum_{k=i+1}^{n-1} a_k \cdot b_{n+i-k}$. Thus $(ab)_{|i}$ is the result of the sum of n products, which explain the origin of n's in the formula.

13.2 Noise variance induced by modulus switching

Let $c = (c_0, c_1)$ a ciphertext of size 2 and modulo q_l. The phase of c is:

$$\phi(c) = c_0 + c_1 \cdot s = \frac{q_l}{t}m + e$$

We denote by v a bound on the variance of coefficients $e_{|i}$ of e. We also denote by v_s the variance of each coefficient of s. Let $q_{l'}$ another modulus of approximately the same size as q_l. The modulus switching of c from modulus q_l to modulus q_l' is given by:

$$c' = \left(\left[\left\lceil \frac{q_{l'}c_0}{q_l} \right\rfloor\right]_{q_l'}, \left[\left\lceil \frac{q_{l'}c_1}{q_l} \right\rfloor\right]_{q_l'}\right)$$

The phase of c' is thus:

$$\begin{aligned}
\phi(c') &= \left\lceil \frac{q_{l'}c_0}{q_l} \right\rfloor + \left\lceil \frac{q_{l'}c_1}{q_l} \right\rfloor \cdot s \\
&= \frac{q_{l'}c_0}{q_l} + \epsilon_0 + \left(\frac{q_{l'}c_1}{q_l} + \epsilon_1\right) \cdot s \\
&\text{with } \epsilon_i \text{ being polynomials with coefficients following } U_{[-\frac{1}{2},\frac{1}{2}]} \\
&= \frac{q_{l'}}{q_l}(c_0 + c_1 \cdot s) + \epsilon_0 + \epsilon_1 \cdot s \\
&= \frac{q_{l'}}{q_l}(\frac{q_l}{t}m + e) + \epsilon_0 + \epsilon_1 \cdot s \\
&= \frac{q_{l'}}{t}m + \frac{q_{l'}}{q_l}e + \epsilon_0 + \epsilon_1 \cdot s
\end{aligned}$$

with $\epsilon_i = \left\lceil \frac{q_{l'}c_i}{q_l} \right\rfloor - \frac{q_{l'}c_i}{q_l}$. The new noise in this ciphertext is thus the polynomial $\frac{q_{l'}}{q_l}e + \epsilon_0 + \epsilon_1 \cdot s$.

We now study the variance of the coefficients of this noise polynomial. Let $i \in [\![0; n-1]\!]$,

$$\begin{aligned}
&\text{Var}\left[\left(\frac{q_{l'}}{q_l}e + \epsilon_0 + \epsilon_1 \cdot s\right)_{|i}\right] \\
&= \left(\frac{q_{l'}}{q_l}\right)^2 v + \frac{1}{12} + \frac{n}{12}v_s
\end{aligned}$$

Since q_l and $q_{l'}$ are approximately of the same size, we can approximate $\frac{q_{l'}}{q_l}$ by 1 leading to the following estimation of the bound on the variance of error coefficients:

$$\mathrm{Var}\left[\Big(\frac{q_{l'}}{q_l}e+\epsilon_0+\epsilon_1\cdot s\Big)_{|i}\right]\approx v+\frac{1}{12}(nv_s+1)$$

13.3 Noise variance induced by multiplication of independant ciphertexts

13.3.1 Average case calculations

Let consider $(b = -as + \frac{q_l}{t}m + e, a) \mod q_l$ and $(b' = -a's + \frac{q_{l'}}{t}m' + e', a') \mod q_{l'}$ two ciphertexts of size 2 encrypting m and m'. We denote by v (resp v') the variance of each coefficient $e_{|i}$ of e (resp. $e'_{|i}$ of e'). We also denote by v_s the variance of each coefficient of s.

The multiplication of $(b = -a\cdot s+\frac{q_l}{t}m+e, a) \mod q_l$ by $(b' = -a'\cdot s+\frac{q_{l'}}{t}m' + e', a') \mod q_{l'}$ outputs a ciphertext structured as a triplet:

$$C=\left(\left[\left\lceil\frac{t}{q_{l'}}b\cdot b'\right\rfloor\right]_{q_l},\left[\left\lceil\frac{t}{q_{l'}}(a\cdot b'+a'\cdot b)\right\rfloor\right]_{q_l},\left[\left\lceil\frac{t}{q_{l'}}a\cdot a'\right\rfloor\right]_{q_l}\right)$$

Then, the phase of C is given by:

$$\begin{aligned}
\phi(C) &= \left\lceil \frac{t}{q_{l'}} b \cdot b' \right\rfloor + \left\lceil \frac{t}{q_{l'}} (a \cdot b' + a' \cdot b) \right\rfloor \cdot s + \left\lceil \frac{t}{q_{l'}} a \cdot a' \right\rfloor \cdot s^2 \\
&= \frac{t}{q_{l'}} b \cdot b' + \epsilon_0 + \Big(\frac{t}{q_{l'}} (a \cdot b' + a' \cdot b) + \epsilon_1 \Big) \cdot s + \Big(\frac{t}{q_{l'}} a \cdot a' + \epsilon_2 \Big) \cdot s^2 \\
&\text{with } \epsilon_i \text{ being polynomials with cofficients following } U_{[-\frac{1}{2},\frac{1}{2}]} \\
&= \frac{t}{q_{l'}} \Big(-a \cdot s + \frac{q_l}{t} m + e \Big) \Big(-a' \cdot s + \frac{q_{l'}}{t} m' + e' \Big) \\
&+ \frac{t}{q_{l'}} \Big(a \cdot (-a' \cdot s + \frac{q_{l'}}{t} m' + e') + a' \cdot (-a \cdot s + \frac{q_l}{t} m + e) \Big) \cdot s \\
&+ \frac{t}{q_{l'}} a \cdot a' \cdot s^2 \\
&+ \epsilon_0 + \epsilon_1 \cdot s + \epsilon_2 \cdot s^2 \\
&= \frac{t}{q_{l'}} \Big(a \cdot a' \cdot s^2 - \frac{q_{l'}}{t} a \cdot m' \cdot s - a \cdot e' \cdot s - \frac{q_l}{t} a' \cdot m \cdot s \\
&\qquad + \frac{q_l}{t} \frac{q_{l'}}{t} m \cdot m' + \frac{q_l}{t} m \cdot e' - a' \cdot e \cdot s + \frac{q_{l'}}{t} e \cdot m' + e \cdot e' \Big) \\
&+ \frac{t}{q_{l'}} \Big(-a \cdot a' \cdot s + \frac{q_l'}{t} a \cdot m' + a \cdot e' - a \cdot a' \cdot s + \frac{q_l}{t} a' \cdot m + a' \cdot e \Big) \cdot s \\
&+ \frac{t}{q_{l'}} a \cdot a' \cdot s^2 \\
&+ \epsilon_0 + \epsilon_1 \cdot s + \epsilon_2 \cdot s^2 \\
&= \frac{t}{q_{l'}} \Big(-\frac{q_{l'}}{t} a \cdot m' \cdot s - a \cdot e' \cdot s - \frac{q_l}{t} a' \cdot m \cdot s \\
&\qquad + \frac{q_l}{t} \frac{q_{l'}}{t} m \cdot m' + \frac{q_l}{t} m \cdot e' - a' \cdot e \cdot s + \frac{q_{l'}}{t} e \cdot m' + e \cdot e' \Big) \\
&+ \frac{t}{q_{l'}} \Big(\frac{q_l'}{t} a \cdot m' \cdot s + a \cdot e' \cdot s + \frac{q_l}{t} a' \cdot m \cdot s + a' \cdot e \cdot s \Big) \\
&+ \epsilon_0 + \epsilon_1 \cdot s + \epsilon_2 \cdot s^2 \\
&= \frac{t}{q_{l'}} \Big(\frac{q_l}{t} \frac{q_{l'}}{t} m \cdot m' + \frac{q_l}{t} m \cdot e' + \frac{q_{l'}}{t} e \cdot m' + e \cdot e' \Big) \\
&+ \epsilon_0 + \epsilon_1 \cdot s + \epsilon_2 \cdot s^2 \\
&= \frac{q_l}{t} m \cdot m' + \frac{q_l}{q_{l'}} m \cdot e' + e \cdot m' + \frac{t}{q_{l'}} e \cdot e' + \epsilon_0 + \epsilon_1 \cdot s + \epsilon_2 \cdot s^2
\end{aligned}$$

The remaining noise in the phase of C is thus: $\frac{q_l}{q_{l'}} m \cdot e' + e \cdot m' + \frac{t}{q_{l'}} e \cdot e' + \epsilon_0 + \epsilon_1 \cdot s + \epsilon_2 \cdot s^2$.

Let us remark that q_l and $q_{l'}$ are two prime of approximately the same size, we will thus approximate $\frac{q_l}{q_{l'}}$ by 1, leading to a noise of $me' + em' + \frac{t}{q_{l'}} ee' + \epsilon_0 + \epsilon_1 \cdot s + \epsilon_2 \cdot s^2$.

We now study the variance of the coefficients of this noise. Let $i \in [\![0; n-1]\!]$,

$$
\begin{aligned}
&\mathrm{Var}\Big[\big(m\cdot e' + e\cdot m' + \frac{t}{q_{l'}}e\cdot e' + \epsilon_0 + \epsilon_1\cdot s + \epsilon_2\cdot s^2\big)_{|i}\Big]\\
&= \mathrm{Var}\Big[\big(m\cdot e'\big)_{|i}\Big] + \mathrm{Var}\Big[\big(e\cdot m'\big)_{|i}\Big] + \mathrm{Var}\Big[\big(\frac{t}{q_{l'}}e\cdot e'\big)_{|i}\Big]\\
&+ \mathrm{Var}\Big[\big(\epsilon_0\big)_{|i}\Big] + \mathrm{Var}\Big[\big(\epsilon_1\cdot s\big)_{|i}\Big] + \mathrm{Var}\Big[\big(\epsilon_2\cdot s^2\big)_{|i}\Big]\\
&= n\frac{t^2-1}{12}v' + n\frac{t^2-1}{12}v + n\frac{t^2}{q_{l'}^2}vv' + \frac{1}{12} + n\frac{1}{12}v_s + n\frac{1}{12}\mathrm{Var}[s^2{}_{|i}]\\
&\approx \frac{nt^2}{12}(v+v') + n\frac{t^2}{q_{l'}^2}vv' + \frac{1}{12}(1 + nv_s + n\mathrm{Var}[s^2{}_{|i}])
\end{aligned}
$$

Since polynomials are modulo $X^n + 1$ we have

$$
s^2{}_{|i} = \sum_{k=0}^{i} s_k s_{i-k} - \sum_{k=i+1}^{n-1} s_k s_{n+i-k}
$$

Remark: since $n = 2^k$ in general, then $i \equiv 1[2]$ implies that there is no square in this product. In that case:

$$
\mathrm{Var}[s^2{}_{|i}] = nv_s^2
$$

Otherwise, if $i \equiv 0[2]$, then there is exactly two squares involved in the sum. We thus have:

$$
\mathrm{Var}[s^2{}_{|i}] = (n-2)v_s^2 + \mathrm{Var}[s_{|\frac{i}{2}}{}^2] + \mathrm{Var}[s_{|\frac{n+i}{2}}{}^2]
$$

In the case of a ternary uniform distribution: $\mathrm{Var}[s_{|i/2}{}^2] = \mathrm{Var}[s_{|\frac{n+i}{2}}{}^2] = \frac{2}{9}$ and $v_s^2 = \left(\frac{2}{3}\right)^2 = \frac{4}{9}$. Thus, nv_s^2 is a valid upper bound for $\mathrm{Var}[s^2{}_{|i}]$ whatever the parity of i.

This leads to the following upper bound for the variance of coefficients in $m\cdot e' + e\cdot m' + \frac{t}{q_{l'}}e\cdot e' + \epsilon_0 + \epsilon_1\cdot s + \epsilon_2\cdot s^2$:

$$
\frac{nt^2}{12}(v+v') + n\frac{t^2}{q_{l'}^2}vv' + \frac{1}{12}(1 + nv_s + n^2v_s^2)
$$

13.3.2 Worst case calculations

We analyze here the variance of the noise after the multiplication of two ciphertexts c and c' with non independant noises. Let v (resp v') be the variance in each coefficient of noise e of ciphertext c (resp. e' of ciphertext c') We recal that the noise after multiplication is : $m\cdot e' + e\cdot m' + \frac{t}{q_{l'}}e\cdot e' + \epsilon_0 + \epsilon_1\cdot s + \epsilon_2\cdot s^2$.

We first analyze the variance of the coefficients of the polynomial ee'. We will use the following inequalities for this purpose.

$$
\mathrm{Cov}(X,Y) \leq \sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)} \qquad (15)
$$

$$\begin{aligned}\mathrm{Var}(\sum_{i=0}^{n-1} X_i) &= \sum_{i=0}^{n-1} \mathrm{Var}(X_i) + \sum_{\substack{i,j=0\\ i\neq j}}^{n-1} \mathrm{Cov}(X_i, X_j)\\ &\leq \sum_{i=0}^{n-1} \mathrm{Var}(X_i) + \sum_{\substack{i,j=0\\ i\neq j}}^{n-1} \sqrt{\mathrm{Var}(X_i)\mathrm{Var}(X_j)}\end{aligned} \tag{16}$$

$$\mathrm{Var}(XY) \leq 2\mathrm{Var}(X)^2\mathrm{Var}(Y)^2 \tag{17}$$

W.l.o.g. we analyze the coefficient $n-1$ of $e \cdot e'$.

$$\begin{aligned}\mathrm{Var}((e\cdot e')_{|n-1}) &= \mathrm{Var}(\sum_{i=0}^{n-1} e_{|i}e'_{|n-1-i})\\ &\leq \sum_{i=0}^{n-1} \mathrm{Var}(e_{|i}e'_{|n-1-i}) + \sum_{\substack{i,j=0\\ i\neq j}}^{n-1} \sqrt{\mathrm{Var}(e_{|i}e'_{|n-1-i})\mathrm{Var}(e_{|j}e'_{|n-1-j})}\\ &\leq \sum_{i=0}^{n-1} 2\mathrm{Var}(e_{|i})^2\mathrm{Var}(e'_{|n-1-i})^2\\ &\quad + \sum_{\substack{i,j=0\\ i\neq j}}^{n-1} \sqrt{4\mathrm{Var}(e_{|i})^2\mathrm{Var}(e'_{|n-1-i})^2\mathrm{Var}(e_{|j})^2\mathrm{Var}(e'_{|n-1-j})^2}\\ &= 2nv^2v'^2 + 2\sum_{\substack{i,j=0\\ i\neq j}}^{n-1} \mathrm{Var}(e_{|i})\mathrm{Var}(e'_{|n-1-i})\mathrm{Var}(e_{|j})\mathrm{Var}(e'_{|n-1-j})\\ &= 2nv^2v'^2 + 2\sum_{\substack{i,j=0\\ i\neq j}}^{n-1} \mathrm{Var}(e_{|i})\mathrm{Var}(e'_{|n-1-i})\mathrm{Var}(e_{|j})\mathrm{Var}(e'_{|n-1-j})\\ &= 2nv^2v'^2 + 2\sum_{\substack{i,j=0\\ i\neq j}}^{n-1} v^2v'^2\\ &= 2nv^2v'^2 + 2n(n-1)v^2v'^2\\ &= 2n^2v^2v'^2\end{aligned} \tag{18}$$

Remark 5 In [8], polynomials v and v' are used to denote the noise in ciphertexts partialy decrypted (before the rounding operation). The link between these polynomials and the noise polynomials e and e' in the cipehrtexts before decryption is given by: $v = \frac{t}{q_l}e$ and $v' = \frac{t}{q_{l'}}e'$. In order to avoid confusion between variances and these terms

v and v', we will rename them z and z'. Thus $z = \frac{t}{q_l}e$ and $z' = \frac{t}{q_{l'}}e'$, which leads to the following formula.

$$\begin{aligned}
\mathrm{Var}\Big(z_{mul}(q_l)_{|i}\Big) &\leq \frac{t^2n^2v_s}{12}\Big(\mathrm{Var}(z_{|i})f(T_1+1) + \mathrm{Var}(z'_{|i})f(T_2+1) \\
&+ 2\sqrt{\mathrm{Var}(z_{|i})\mathrm{Var}(z'_{|i})f(T_1+1)f(T_2+1)}\Big) + \mathrm{Var}\Big((z\cdot z')_{|i}\Big) \\
&= \frac{t^2n^2v_s}{12}\Big(\mathrm{Var}(\frac{t}{q_l}e_{|i})f(T_1+1) + \mathrm{Var}(\frac{t}{q_{l'}}e'_{|i})f(T_2+1) \\
&+ 2\sqrt{\mathrm{Var}(\frac{t}{q_l}e_{|i})\mathrm{Var}(\frac{t}{q_{l'}}e'_{|i})f(T_1+1)f(T_2+1)}\Big) \\
&+ \mathrm{Var}\Big((\frac{t}{q_l}e\cdot\frac{t}{q_{l'}}e')_{|i}\Big) \\
&= \frac{t^2n^2v_s}{12}\Big(\frac{t^2}{q_l^2}\mathrm{Var}(e_{|i})f(T_1+1) + \frac{t^2}{q_{l'}^2}\mathrm{Var}(e'_{|i})f(T_2+1) \\
&+ 2\sqrt{\frac{t^2}{q_l^2}\mathrm{Var}(e_{|i})\frac{t^2}{q_{l'}^2}\mathrm{Var}(e'_{|i})f(T_1+1)f(T_2+1)}\Big) \\
&+ \frac{t^2}{q_l^2}\frac{t^2}{q_{l'}^2}\mathrm{Var}\Big((e\cdot e')_{|i}\Big) \\
&= \frac{t^2n^2v_s}{12}\Big(\frac{t^2}{q_l^2}\mathrm{Var}(e_{|i})f(T_1+1) + \frac{t^2}{q_{l'}^2}\mathrm{Var}(e'_{|i})f(T_2+1) \\
&+ 2\frac{t}{q_l}\frac{t}{q_{l'}}\sqrt{\mathrm{Var}(e_{|i})\mathrm{Var}(e'_{|i})f(T_1+1)f(T_2+1)}\Big) \\
&+ \frac{t^2}{q_l^2}\frac{t^2}{q_{l'}^2}\mathrm{Var}\Big((e\cdot e')_{|i}\Big)
\end{aligned} \tag{19}$$

Remark 6 The quantity $z_{mul}(q_l)_{|i}$ is the value of the noise rescaled in the plaintext space. In our case, we are interested in the noise in the ciphertext space, thus $(e_{mul})_{|i} = \frac{q_l}{t}z_{mul}(q_l)_{|i}$. We can equivalently write $z_{mul}(q_l)_{|i} = \frac{t}{q_l}(e_{mul})_{|i}$

Combining the previous remark 6 and inequality (19), we have:

$$\begin{aligned}
\mathrm{Var}\Big(\frac{t}{q_l}(e_{mul})_{|i}\Big) &\leq \frac{t^2n^2v_s}{12}\Big(\frac{t^2}{q_l^2}\mathrm{Var}(e_{|i})f(T_1+1) + \frac{t^2}{q_{l'}^2}\mathrm{Var}(e'_{|i})f(T_2+1) \\
&+ 2\frac{t}{q_l}\frac{t}{q_{l'}}\sqrt{\mathrm{Var}(e_{|i})\mathrm{Var}(e'_{|i})f(T_1+1)f(T_2+1)}\Big) \\
&+ \frac{t^2}{q_l^2}\frac{t^2}{q_{l'}^2}\mathrm{Var}\Big((e\cdot e')_{|i}\Big)
\end{aligned}$$

and thus

$$\begin{aligned}\frac{t^2}{q_l^2}\mathrm{Var}\Big((e_{mul})_{|i}\Big) \leq \frac{t^2 n^2 v_s}{12}&\Bigg(\frac{t^2}{q_l^2}\mathrm{Var}(e_{|i})f(T_1+1)+\frac{t^2}{q_{l'}^2}\mathrm{Var}(e'_{|i})f(T_2+1)\\&+2\frac{t}{q_l}\frac{t}{q_{l'}}\sqrt{\mathrm{Var}(e_{|i})\mathrm{Var}(e'_{|i})f(T_1+1)f(T_2+1)}\Bigg)\\&+\frac{t^2}{q_l^2}\frac{t^2}{q_{l'}^2}\mathrm{Var}\Big((e\cdot e')_{|i}\Big)\end{aligned}$$

rewritten

$$\begin{aligned}\mathrm{Var}\Big((e_{mul})_{|i}\Big) &\leq \frac{t^2 n^2 v_s}{12}\Bigg(\mathrm{Var}(e_{|i})f(T_1+1)+\frac{q_l^2}{t^2}\frac{t^2}{q_{l'}^2}\mathrm{Var}(e'_{|i})f(T_2+1)\\&+2\frac{q_l^2}{t^2}\frac{t}{q_l}\frac{t}{q_{l'}}\sqrt{\mathrm{Var}(e_{|i})\mathrm{Var}(e'_{|i})f(T_1+1)f(T_2+1)}\Bigg)\\&+\frac{q_l^2}{t^2}\frac{t^2}{q_l^2}\frac{t^2}{q_{l'}^2}\mathrm{Var}\Big((e\cdot e')_{|i}\Big)\\&=\frac{t^2 n^2 v_s}{12}\Bigg(\mathrm{Var}(e_{|i})f(T_1+1)+\Big(\frac{q_l}{q_{l'}}\Big)^2\mathrm{Var}(e'_{|i})f(T_2+1)\\&+2\frac{q_l}{q_{l'}}\sqrt{\mathrm{Var}(e_{|i})\mathrm{Var}(e'_{|i})f(T_1+1)f(T_2+1)}\Bigg)\\&+\frac{t^2}{q_{l'}^2}\mathrm{Var}\Big((e\cdot e')_{|i}\Big)\\&\approx\frac{t^2 n^2 v_s}{12}\Bigg(\mathrm{Var}(e_{|i})f(T_1+1)+\mathrm{Var}(e'_{|i})f(T_2+1)\\&+2\sqrt{\mathrm{Var}(e_{|i})\mathrm{Var}(e'_{|i})f(T_1+1)f(T_2+1)}\Bigg)+\frac{t^2}{q_{l'}^2}\mathrm{Var}\Big((e\cdot e')_{|i}\Big)\end{aligned}$$

Applying inequality (18) finally gives:

$$\begin{aligned}\mathrm{Var}\Big((e_{mul})_{|i}\Big) \leq \frac{t^2 n^2 v_s}{12}&\Bigg(vf(T_1+1)+v'f(T_2+1)\\&+2\sqrt{vv'f(T_1+1)f(T_2+1)}\Bigg)+\frac{t^2}{q_{l'}^2}2n^2v^2v'^2\end{aligned}\tag{20}$$

13.4 Noise variance induced by relinearization of (triplet-structured) ciphertexts

Let us recall the elements of the keyswitch. P is an integer greater than q and of approximately the same size.

KSGen$^{\text{GHS}}(s, s^2)$: $a' \leftarrow \mathcal{U}_{qP}$, $e' \leftarrow \chi_e$ and set

$$\mathbf{ks}^{\text{GHS}} = (b', a') = \left(\left[Ps^2 - a' \cdot s + e' \right]_{qP}, a' \right)$$

Let $d = (d_0, d_1, d_2)$ a ciphertext of size 3 and modulo q_l. The phase of d is:

$$\phi(d) = d_0 + d_1 \cdot s + d_2 \cdot s^2 = \frac{q_l}{t} m + e$$

We denote by v (resp. v') a bound on the variance of coefficients $e_{|i}$ of e (resp. $e'_{|i}$ of e'). We also denote by v_s the variance of each coefficient of s.

Remark 7 As e' is a noise picked at random following the distribution used to generate fresh noises, v' is the variance of a fresh noise.

The relinearization of d is:

$$r = \left(\left[d_0 + \left\lceil \frac{d_2 \cdot b'}{P} \right\rfloor \right]_{q_l}, \left[d_1 + \left\lceil \frac{d_2 \cdot a'}{P} \right\rfloor \right]_{q_l} \right)$$

Computing the phase of r gives:

$$\begin{aligned}
\phi(r) &= d_0 + \left\lceil \frac{d_2 \cdot b'}{P} \right\rfloor + \left(d_1 + \left\lceil \frac{d_2 \cdot a'}{P} \right\rfloor \right) \cdot s \\
&= d_0 + \frac{d_2 \cdot b'}{P} + \epsilon_0 + \left(d_1 + \frac{d_2 \cdot a'}{P} + \epsilon_1 \right) \cdot s \\
&\quad \text{with } \epsilon_i \text{ being polynomials with cofficients following } U_{[-\frac{1}{2}, \frac{1}{2}]} \\
&= d_0 + d_2 \cdot \frac{P \cdot s^2 - a' \cdot s + e'}{P} + \epsilon_0 + \left(d_1 + d_2 \cdot \frac{a'}{P} + \epsilon_1 \right) \cdot s \\
&= d_0 + d_2 \cdot (s^2 - \frac{a' \cdot s}{P} + \frac{e'}{P}) + \epsilon_0 + d_1 \cdot s + d_2 \cdot \frac{a' \cdot s}{P} + \epsilon_1 \cdot s \\
&= d_0 + d_1 \cdot s + d_2 \cdot s^2 + \frac{d_2 \cdot e'}{P} + \epsilon_0 + \epsilon_1 \cdot s \\
&= \frac{q_l}{t} m + e + \frac{d_2 \cdot e'}{P} + \epsilon_0 + \epsilon_1 \cdot s
\end{aligned}$$

The noise *added* by the relinearization is thus:

$$\frac{d_2 \cdot e'}{P} + \epsilon_0 + \epsilon_1 \cdot s$$

We now study the variance of the coefficients of this new *added* noise. Let $i \in [\![0; n-1]\!]$,

$$\begin{aligned}
&\mathrm{Var}\Big[\Big(\frac{d_2 \cdot e'}{P} + \epsilon_0 + \epsilon_1 \cdot s\Big)_{|i}\Big] \\
&= \frac{1}{P^2}\mathrm{Var}\Big[d_2 \cdot e'\Big]_{|i} + \mathrm{Var}[\epsilon_{0\ |i}] + \mathrm{Var}\Big[(\epsilon_1 \cdot s)_{|i}\Big] \\
&= \frac{nq^2}{12 \cdot P^2} v' + \frac{1}{12} + \frac{n}{12} v_s \\
&\leq \frac{n}{12} v' + \frac{1}{12} + \frac{n}{12} v_s \\
&= \frac{1}{12}\Big(n(v' + v_s) + 1\Big)
\end{aligned}$$

14 Illustration of the three parties protocol of section 6

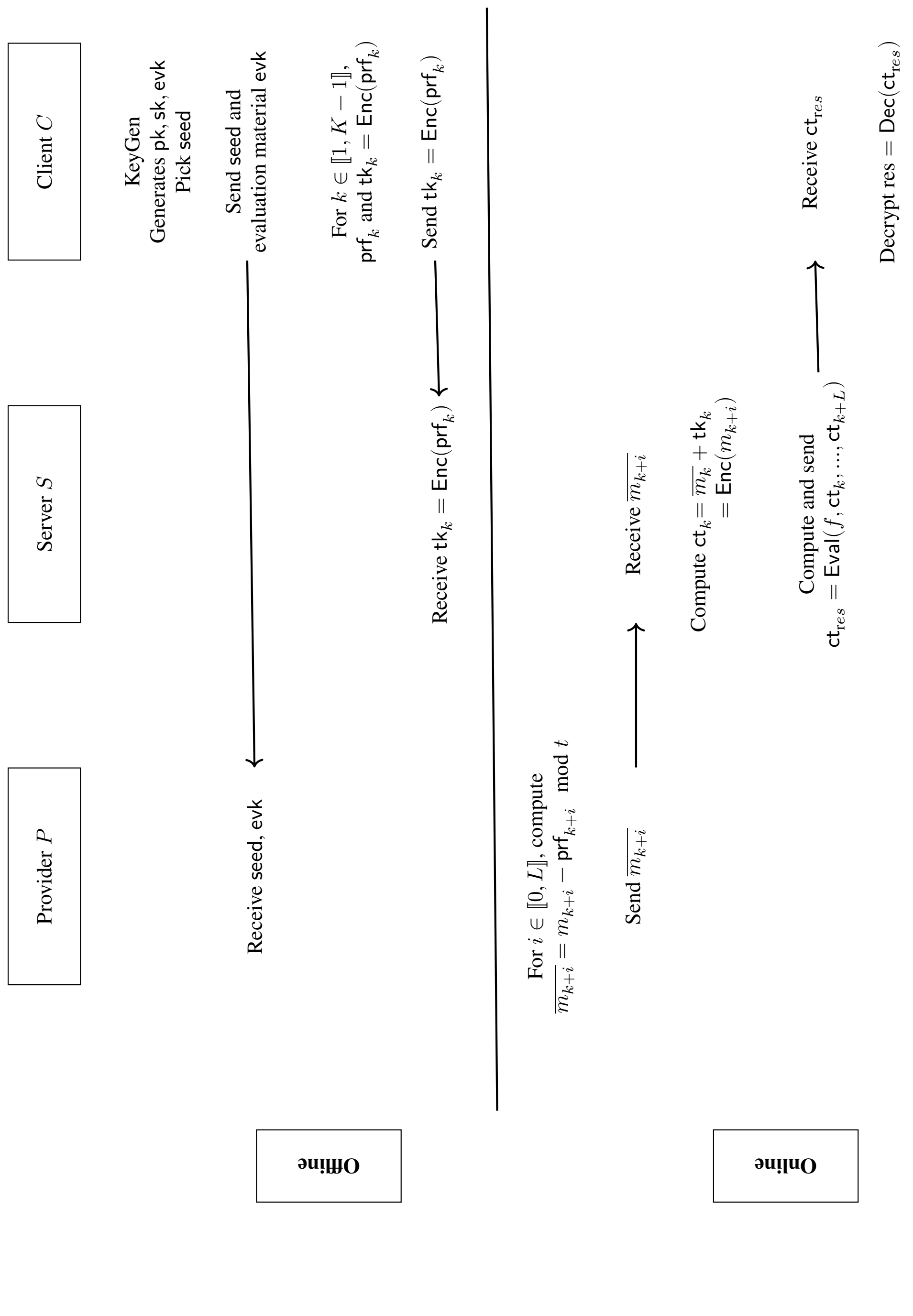
Client C
Server S
Provider P
KeyGen
Generates $\mathsf{pk}, \mathsf{sk}, \mathsf{evk}$
Pick seed
Send seed and evaluation material evk
Receive seed, evk
For $k \in [\![1, K-1]\!]$, prf_k and $\mathsf{tk}_k = \mathsf{Enc}(\mathsf{prf}_k)$
Send $\mathsf{tk}_k = \mathsf{Enc}(\mathsf{prf}_k)$
Receive $\mathsf{tk}_k = \mathsf{Enc}(\mathsf{prf}_k)$
Offline
For $i \in [\![0, L]\!]$, compute $\overline{m_{k+i}} = m_{k+i} - \mathsf{prf}_{k+i} \mod t$
Send $\overline{m_{k+i}}$
Receive $\overline{m_{k+i}}$
Compute $\mathsf{ct}_k = \overline{m_k} + \mathsf{tk}_k = \mathsf{Enc}(m_{k+i})$
Compute and send $\mathsf{ct}_{res} = \mathsf{Eval}(f, \mathsf{ct}_k, ..., \mathsf{ct}_{k+L})$
Receive ct_{res}
Decrypt $\mathsf{res} = \mathsf{Dec}(\mathsf{ct}_{res})$
Online

Fig. 1 Diagram of our protocol against malicious providers in a private key setting