Impossibility of CPA^D security for a class of FHE schemes*

Marina Checri, Pierre-Emmanuel Clet, Marc Renard and Renaud Sirdey

Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France,
marina.checri.pro@proton.me, pierre-emmanuel.clet@cea.fr,
marc.renard.pro@proton.me, renaud.sirdey@cea.fr

Abstract. In this paper, we focus on the class of *at least* linearly homomorphic schemes such that their homomorphic addition operator is itself a linear operator over the ciphertext domain. This class of schemes, which we refer to as HELLHO schemes, notably encompasses the basic variant of most practically used FHE schemes such as BFV, BGV, CKKS and TFHE, *as long as their mode of operation allows for homomorphic additions not immediately followed by bootstrapping*, but also several extensions of these as well as other less mainstream proposals based on other assumptions. Although the aforementioned specific FHE are known to be CPA^D insecure due to the existence of concrete attacks against them in that model, we first show by a simple argument that no HELLHO scheme can achieve CPA^D security. Moving one step further, we also establish several counter-intuitive facts for the class of HELLHO schemes: for example that any CCA2 attack can be turned into a CPA^D one or that, still *only* for this class of schemes, CCA1 security is equivalent to CPA$^D_1$ security (a strictly weaker "CCA1-style" variant of CPA^D). Among other practical consequences, the results in this paper notably allow to show that instantiating the Dynamic Error Estimation (DEE) heuristic of Li et al. (Crypto'22) from *any* "natural" (R)LWE-based schemes cannot yield a CPA^D secure scheme. As another notable consequence, we further exhibit a practical KR^D attack on the DE-CKKS scheme, which is presently supported in OpenFHE. We conclude the paper by experimental results showing that this attack is able to perform a full key recovery on DE-CKKS in a matter of hours on an average laptop PC.

Keywords: Homomorphic encryption · Impossibility results · CPA^D security.

1 Introduction

Following Li and Micciancio's seminal paper introducing the notion [46], CPA^D security has established itself as a foundational notion for the security of FHE,

* This work was supported by the France 2030 ANR Projects ANR-22-PECY-003 SecureCompute.

revealing a subtle interplay between correctness after homomorphic evaluation and security in the beyond-CPA regime.

In a nutshell, CPA^D security takes a first step towards providing the adversary with a decryption oracle but does so only under the stringent constraint that the adversary may ask for the decryption only of well-formed ciphertexts, i.e. well-formed fresh encryptions or obtained by performing genuine homomorphic evaluations over well-formed fresh encryptions. Whether CPA^D security matters depends heavily on whether the adversary can observe decryption errors. In particular, the notion of CPA^D security is *not* relevant for schemes achieving unconditional correctness after evaluation: since the adversary can compute all outputs of the decryption oracle on its own, the notion is then equivalent to CPA security. It is important to emphasize that, under the CPA^D model in which full control is given to the adversary on the homomorphic calculations that are performed, the adversary is *not* constrained to limit its evaluations to a prescribed class of circuits (other than that of size polynomial in the security parameter). Under such circumstances, the basic variants of almost all CPA-secure FHE used in practice do not achieve correctness after evaluation at least when their mode of operation does not include bootstrapping. This is true for CKKS, due to its approximate nature, but also for the other "exact" schemes such as BFV or BGV: homomorphic calculations increase the magnitude of the LWE noise in the ciphertexts, giving the adversary a means to trigger decryption errors. It then turns out that this is enough to practically break all these schemes in the CPA^D model [46,44,30,31] as those errors eventually leak the LWE noises[1].

In this paper, we reveal that there are fundamental reasons, *independent of the LWE assumption*, why these schemes do not achieve CPA^D security. Indeed, we focus on the class of *at least* linearly homomorphic schemes with a homomorphic addition operator that is linear over their ciphertext domain. We coin them HELLHO schemes for "Homomorphic Encryption schemes with Linear Linearly Homomorphic Operators". This class of schemes includes most LWE-based or RLWE-based FHE schemes such as BFV, BGV, CKKS and others, in their somewhat homomorphic or levelled basic variants, as well as some natural extensions of these that we will consider. It also covers a number of less popular proposals based on codes or on LIP. Our main result is to show that *any* CPA secure FHE with a linear homomorphic addition operator cannot achieve CPA^D security under unrestricted evaluation. As a byproduct, we notably rule out that such schemes can both achieve unconditional correctness after evaluation and be CPA secure. We also show that, for this class of schemes, CCA1 security coincides with CPA_1^D security (a strictly weaker variant of CPA^D security consistent with CCA1's decryption oracle early closure). Additionally, we consider CCA security for FHE and show that it is also impossible for HELLHO schemes to achieve CCA security notions such as the semi-active security (IND-SA) notion of [63], IV-CCA [65], vCCA [52] and others. Overall, our impossibility results

[1] Even in the case of a mode of operation with systematic bootstrapping, CPA^D attacks are still possible [31] unless the scheme parameters are chosen to achieve a negligible probability of bootstrapping error.

turn into cul-de-sacs several avenues for building CPA^D or CCA secure FHE that may have appeared legitimate. This is in particular the case for the Dynamic Error Estimation (DEE) heuristic[2] that was applied to CKKS in [47]. Although this heuristic was shown in that paper not to achieve CPA^D security in the case of CKKS, we further establish that it is also the case when it is instantiated in the BFV/BGV setting (i.e. with message scaling/rounding at encryption/decryption). More importantly, in the weaker KR^D model, our results also yield *practical* full key recovery attacks against the resulting schemes, DE-RLWE and DE-CKKS, the latter of which being presently supported in OpenFHE.

We also emphasize that this is not the first time that linearity is a liability in the context of homomorphic encryption. For example, it is shown in [22] that any scheme with a linear decryption function (as, for example, in LPN-based schemes) cannot achieve CPA security as soon as it is homomorphic-enough to evaluate a majority voting operator. Our results depart from this as, in our case, the liability comes from the linearity of the homomorphic addition operator rather than that of the decryption function on which we place no constraint.

Regarding CCA security for FHE, it may be interesting to highlight that the large majority of works investigating either or both CCA security notions achievable by FHE or FHE constructions achieving some degree of CCA security [17,27,52,65] do so under strong correctness assumptions. This is to the notable exception of [49][3], [29] (in which only a weaker variant of CCA1 is achieved for an LWE-based linear-only HE) as well as [25], which investigates how to leverage CPA^D secure FHE to build CCA secure ones. The present work sheds some new light on the structure of the constructions that this latter path may yield.

1.1 Overview of our contributions

The contributions of this paper are as follows:

- We introduce the notion of $\mathbb{F}$-HELLHO and HELLHO FHE schemes:
  - In a nutshell, an $\mathbb{F}$-HELLHO scheme is an *at least* linearly homomorphic scheme with ciphertext domain $\mathbb{Z}_p^N$, for $N \leq O(\text{poly}(\lambda))$ and a prime $p \leq 2^{O(\text{poly}(\lambda))}$, such that its homomorphic addition operator yields a vector space of dimension N over $\mathbb{Z}_p$ and such that any set of N well-formed fresh ciphertexts forms a basis of this vector space with non-negligible probability.
  - The notion of HELLHO scheme then generalizes the notion of $\mathbb{F}$-HELLHO scheme when the homomorphic addition only yields a $\mathbb{Z}$-module for which $O(\text{poly}(\lambda))$ well-formed fresh ciphertexts form a generating set with non-negligible probability and such that, given a generating set, linear combinations can be efficiently found.

[2] This heuristic consists in using a number of message-less slots in a RLWE ciphertext to obtain an estimation of the ciphertext's noise variance that is then used for noise flooding at decryption.

[3] However, the assumption underlying the CPA security of the construction in [49] is subject to caution following results in [37,14] and the knowledge assumption used in that paper is not secure against a quantum adversary [38,39].

– We argue that the class of HELLHO schemes includes the basic variants of most of the lattice-based FHE (notably BGV, BFV, CKKS, TFHE and others) *as soon as their mode of operation allows unrestricted use of the natural homomorphic addition*, e.g. without an immediate follow-up bootstrapping (for most of them, this is the usual setup). The definition also covers other less mainstream proposals, for example, based on codes or on the LIP assumption.
– For HELLHO schemes, we then establish the fact that *any element of the ciphertext domain can be efficiently obtained from a sufficiently large set of well-formed ciphertexts (e.g. fresh well-formed encryptions of 0) by means only of legitimate homomorphic operations*. As a consequence, we will be able to *insert (or re-insert) any element of the ciphertext domain into the internal state of a CPA^D challenger such that it sees it as an encryption of 0 and thus accepts a decryption request over it*. This yields several results, some of them counter-intuitive, among which,
  • HELLHO schemes cannot achieve CPA^D security.
  • HELLHO schemes cannot achieve statistical correctness after evaluation (a notion which we denote $\mathcal{F}(\lambda)$-correctness).
  • Although we do not rule out that CPA$^D_1$ (Sect. 2.2) or CCA1 secure HELLHO schemes may exist, we establish that, for this class of schemes, these two latter security notions are equivalent.
– We then turn our attention to the recent CCA security notions for FHE and show, perhaps unsurprisingly, that HELLHO schemes cannot achieve IND-SA, IV-CCA, vCCA or vCCA^D security. Because of the intricate pattern of relations among CPA^D and CCA security notions for FHE (see Fig. 1), some of these results require specific proofs.
– We finally investigate some practical consequences, in particular with respect to FHE schemes designed following the Dynamic Error Estimation (DEE) heuristic that Li et al. [47] introduced in the context of CKKS. In a nutshell, this heuristic consists in using a number of message-less slots in an RLWE ciphertext to perform an estimation of the ciphertext noise variance at decryption. The intent is then to use this estimation to generate a smudging/flooding noise in an attempt to avoid leakage on the ciphertext noise by the decryption function (and, hence, achieve CPA^D security). We show that both the DE-CKKS scheme of [46] and its natural adaptation to the usual RLWE setting, DE-RLWE, are both HELLHO schemes and therefore cannot achieve CPA^D security. *Note that the CPA^D (in)security of DE-CKKS was already established in [47] but using a completely different line of attack exploiting a "mild" leakage on the norm of the cleartext inputs to a homomorphic calculation*. Then, [47] explicitly left open the question of the possibility of more harmful leakage that we positively answer.
– Lastly, we turn our attention to the KR^D security of both DE-RLWE and DE-CKKS (the possible KR^D security of DE-CKKS was also left as an interesting open problem in [47]). Using our results (and in particular the CCA1/CPA$^D_1$ equivalence that we have established for HELLHO schemes), we demonstrate that both schemes can be practically broken in the KR^D

model. This is more easily done for DE-RLWE by leveraging on a folklore CCA1(-turned-CPA$_1^D$) attack against RLWE that we adapt such that it remains unaffected (with large probability) by the additional smudging performed at decryption. For DE-CKKS, which is currently supported in OpenFHE, we design a more involved KR^D attack and conclude the paper by experimental results showing that it is able to perform a full key recovery in a matter of hours on an average laptop PC.

For convenience, we provide in Fig. 1 a summary of the relationships among the security notions considered in this paper.

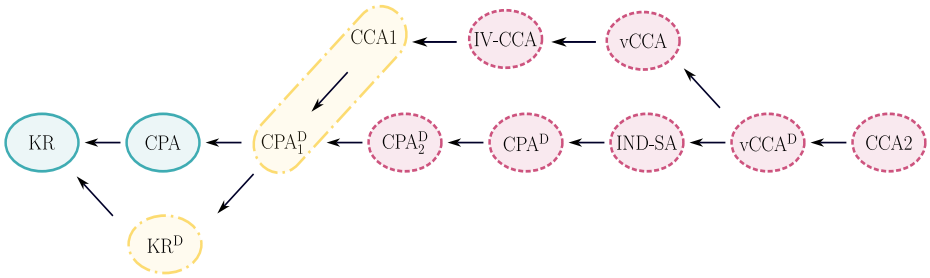

Fig. 1. Overview of the relations among the security notions considered in this paper. The definition of HELLHO schemes itself implies that they satisfy the notions in green, however, they cannot achieve the notions in red. The notions in yellow are not ruled out by the results in this paper, although we show that CPA$_1^D$ is equivalent to CCA1 for this special class of schemes, and we also show some specific HELLHO schemes insecure in both this model and the KR^D one.

1.2 Paper organization

This paper is organized as follows. Section 2 covers the preliminaries on correctness, security notions and smudging. Then, Sect. 3 provides the definitions of HELLHO and $\mathbb{F}$-HELLHO schemes and gives their first properties. Sect. 4 subsequently provides our impossibility results for HELLHO schemes regarding both CPA^D and CCA security notions. Finally, Sect. 5 digs into the practical consequences of our results for schemes embedding dynamic error estimation, Sect. 6 briefly concludes the paper. Sect. 7 summarizes the outcome of Responsible Disclosure.

2 Preliminaries

Given $l, u \in \mathbb{Z}^2$, 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.

Given two discrete random variables X and Y we write $X \stackrel{i}{=} Y$ when the distribution of X and that of Y are such that $d(f_X, f_Y) \leq \mathrm{negl}(\lambda)$ where

$$d(f_X, f_Y) = \frac{1}{2} \sum_{k=-\infty}^{+\infty} |P(X = k) - P(Y = k)|$$

is the usual statistical distance. In this case, X and Y are said to be *statistically indistinguishable* or, for short, *indistinguishable* from one another. For simplicity sake, we write λ to denote both the computational security parameter *and* the statistical security parameter.

2.1 Correctness properties

We define an encryption scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ over key space $\mathcal{K}$, plaintext domain $\mathcal{P}$ and ciphertext domain $\mathcal{C}$[4] as a triplet of PPT algorithms:

- KeyGen: on input 1^λ, outputs an encryption key ek and a decryption key sk.
- Enc: on input $m \in \mathcal{P}$ and ek, outputs an encryption $c \in \mathcal{C}$ of m.
- Dec: on input $c \in \mathcal{C}$ and sk, outputs a decryption $m \in \mathcal{P} \cup \{\perp\}$ of c.

Let COIN denote the randomness space of $\mathcal{E}$. We sometimes externalize the randomness used in the encryption function by means of the notation $\mathsf{Enc}(m; r)$, with $m \in \mathcal{P}$ and $r \in \mathsf{COIN}$. In this latter case, the function $\mathsf{Enc} : \mathcal{P} \times \mathsf{COIN} \longrightarrow \mathcal{C}$ is deterministic[5]. When ek is public, we say that $\mathcal{E}$ is a *public-key* encryption scheme *and use* pk *to denote* ek. When for all $(\mathsf{ek}, \mathsf{sk}) \in \mathcal{K}$, all $m \in \mathcal{P}$ and $r \stackrel{\$}{\leftarrow} \mathsf{COIN}$ we have that,

$$\Pr(\mathsf{Dec}(\mathsf{Enc}(m; r)) \neq m) \leq \mathrm{negl}(\lambda), \tag{1}$$

we say that $\mathcal{E}$ is *statistically correct* or simply *correct*[6] (remark that a decryption error occurs systematically when the decryption function returns $\perp$, since $\perp \notin \mathcal{P}$). When the above probability is zero, we talk of *perfect* correctness. Lastly, when for all $(\mathsf{ek}, \mathsf{sk}) \in \mathcal{K}$, all $m \in \mathcal{P}$ and $r \stackrel{\$}{\leftarrow} \mathsf{COIN}$,

$$\Pr(\mathsf{Dec}(\mathsf{Enc}(m; r)) = m) \geq \frac{3}{4}, \tag{2}$$

we say that $\mathcal{E}$ is *loosely correct*[7].

Let $\mathcal{O}$ denote a set of operations and let $\mathcal{F}(\lambda)$ denote the set of circuits over $\mathcal{O}$ of polynomially bounded size in the security parameter λ. We define a

[4] In many cases, $\mathcal{C}$ and possibly $\mathcal{P}$ are dependent on the security parameter λ but we omit this dependency in our notations.

[5] In this case we assume that r is uniformly picked in COIN to provide a seed that the encryption function then uses to sample from the proper distributions.

[6] There are additional definitions for correctness obtained when taking the probability on the coin space for the key generation [40]. We do not consider these in this paper.

[7] Intuitively, we use this quite weak notion as a bare minimum requirement for a scheme to be useful. Indeed, one should legitimately expect any useful scheme to achieve a much larger probability of correct fresh ciphertext decryption.

homomorphic encryption (HE) scheme over $\mathcal{F}(\lambda)$, $\mathcal{E}_H$, as an encryption scheme augmented by a *deterministic*[8] polynomial-time algorithm Eval which, on input $f \in \mathcal{F}(\lambda)$ and $c_0, ..., c_{L-1} \in \mathcal{C}^L$, where L denotes the arity of function f, outputs a new *evaluated* ciphertext. Let $\mathcal{H} \subseteq \mathcal{F}(\lambda)$, when $\mathcal{E}_H$ satisfies condition (1) and when Eval is such that for all $(\mathsf{ek}, \mathsf{sk}) \in \mathcal{K}$, all $f \in \mathcal{H}$, all $m_0, ..., m_{L-1} \in \mathcal{P}^L$ and $(r_0, ..., r_{L-1}) \xleftarrow{\$} \mathsf{COIN}^L$,

$$\Pr(\mathsf{Dec}(c_e) \neq f(m_0, ..., m_{L-1})) \leq \mathrm{negl}(\lambda), \tag{3}$$

with,

$$c_e = \mathsf{Eval}(f, \mathsf{Enc}(m_0; r_0), ..., \mathsf{Enc}(m_{L-1}; r_{L-1})),$$

we say that $\mathcal{E}_H$ is a *statistically correct* or simply *correct* HE scheme *over* $\mathcal{H}$ or that $\mathcal{E}_H$ is $\mathcal{H}$*-correct*. Consistently with above, when the probability in Eqs. (1) and (3) is 0, we say that $\mathcal{E}_H$ is *perfectly correct* over $\mathcal{H}$ or *perfectly* $\mathcal{H}$*-correct*. When Eqs. (1) and (3) do not hold, we say that $\mathcal{E}_H$ is an *approximate* HE scheme. Consistently with [47], to avoid arbitrary schemes with unreliable Eval to be marketed as approximate HE schemes, we add an additional condition that, for some (small) $\varepsilon \geq 0$, the following holds

$$\Pr(\|\mathsf{Dec}(c_e) - f(m_0, ..., m_{L-1})\|_\infty \leq \varepsilon) \geq \mu, \tag{4}$$

with $\mu \geq \frac{3}{4}$. Lastly, a scheme that satisfies Eq. (2) and Eq. (4) with $\varepsilon = 0$ and $\mu \geq \frac{3}{4}$ is said to be *somewhat or* μ*-somewhat correct over* $\mathcal{H}$.

To conclude this section, we highlight that $\mathcal{F}(\lambda)$-correctness or even $\mathcal{H}$-correctness, for some $\mathcal{H} \subset \mathcal{F}(\lambda)$, are quite strong properties. Clearly, for a given application scenario, a scheme achieving $(1 - 2^{-40})$-somewhat correctness over some small $\mathcal{H} \subset \mathcal{F}(\lambda)$ that covers the needed functionalities is practically useful.

2.2 Security notions

CPA^D security (and KR^D, CPA$^D_1$, CPA$^D_2$ security). The CPA^D game has been introduced in the context of approximate FHE [46]. CPA^D security is an extension of CPA security (recalled in App. E.1) defined as follows.

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

- Encryption oracle (E): given $m \in \mathcal{P}$ and ek compute $c = \mathsf{Enc}(m, \mathsf{ek})$, return c after updating $S := [S; (m, m, c)]$.
- Challenge oracle (C): given $m_0, m_1 \in \mathcal{P}$ $(m_0 \neq m_1)$ and ek, compute $c = \mathsf{Enc}(m_\gamma, \mathsf{ek})$, return c after doing $S := [S; (m_0, m_1, c)]$.

[8] As is the case for the basic variants of mainstream FHE schemes such as BFV, BGV, TFHE and even CKKS.

– Evaluation oracle (H): given $f \in \mathcal{F}(\lambda)$ and $l_0, \ldots, l_{L-1}$ ($l_i \in [\![0, |S| - 1]\!], \forall i$), compute, $m'_i = f(S[l_0].m_i, \ldots, S[l_{L-1}].m_i)$, $i \in \{0, 1\}$, as well as,

$$c' = \mathsf{Eval}(f, S[l_0].c, \ldots, S[l_{L-1}].c),$$

return c' after doing, $S \coloneqq [S; (m'_0, m'_1, c')]$.
– Decryption oracle (D): given $l \in [\![0, |S| - 1]\!]$ and sk, proceed as follows. If $S[l].m_0 \neq S[l].m_1$, return $\perp$. Otherwise return $\mathsf{Dec}(S[l].c, \mathsf{sk})$.

With these oracles, the CPA^D experiment is,

$$\begin{aligned}
&(\mathsf{ek}, \mathsf{sk}) \leftarrow \mathcal{E}_H.\mathsf{KeyGen}(1^\lambda),\\
&\gamma \xleftarrow{\$} \{0, 1\},\\
&\gamma' \leftarrow \mathcal{A}^{\mathsf{E}(.,\mathsf{ek}),\mathsf{C}(.,.,\mathsf{ek}),\mathsf{H}(\ldots),\mathsf{D}(.,\mathsf{sk})}([\mathsf{ek}]),\\
&\gamma = \gamma'.
\end{aligned}$$

where ek is provided only in the public-key case, and $\mathsf{E}(., \mathsf{ek})$ is always provided.

Remark that the decryption oracle accepts only indices that point to ciphertexts in the game state which are necessarily *well-formed*, i.e. either 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 ciphertexts from the game state (hence well-formed). As such, the above game does not capture any CCA aspect. Let us also emphasize that, in the above game, $\mathcal{A}$ *has full control over the (polynomial time) homomorphic calculations performed* as f is included in the evaluation request parameters.

In addition to the above multiple-challenge notion initially defined in [46], we will also consider its following weaker restrictions:

– CPA$_2^D$ [25]: restriction of CPA^D to the single challenge case where the adversary is allowed only one request to the challenge oracle C.
– CPA$_1^D$ [25]: restriction of CPA$_2^D$ with the decryption oracle closing after the unique challenge request (similar in spirit to the CCA1/CCA2 definitions).
– KR^D [46]: a challenge-less variant of CPA^D in which the adversary wins the game when it retrieves the secret decryption key.

With respect to these latter notions, [25] has established the following separation results in the general regime where approximate or somewhat correct FHE schemes are considered:

$$\text{CPA} \prec \text{CPA}_1^D \prec \text{CPA}_2^D \prec \text{CPA}^D. \tag{5}$$

Note that CPA$_1^D$ is different from non-adaptive CPA^D as defined and studied in [46] (also referred to as CPA$_0^D$ in [29]). Indeed, there is a slight difference between the notion of adaptability as understood in the multiple-challenge context of [46] (the adversary specifies all its requests straight after seeing the public material of the scheme, i.e. no adaptability to the oracles' responses) and that which is usually assumed between single-challenge CCA1 and CCA2 (the adversary

performs all its decryption requests before the *unique* challenge ciphertext is published, i.e. no adaptability to the challenge ciphertext). On the other side of the spectrum, CPA^D security also has a strictly stronger variant, sCPA^D, in which the encryption randomness is under the adversary's control [13], i.e. the adversary gets to choose the randomness used in encryption but *not* in challenge requests in the above game.

For completion, the definitions for CPA, CCA1 and CCA2 security are recalled in App. E.1. In the general regime where $\mathcal{F}(\lambda)$-correctness does not hold, we have $\text{CPA} \prec \text{CPA}_1^D \prec \text{CCA1}$ (see [25] again).

As a last important point, we emphasize that the original definition of CPA^D security [46,47] does not put any restriction on the set in which the PPT adversary is allowed to choose the functions that it submits to the evaluation oracle. For example, recall the notations for Eq. (3) and consider an HE scheme over $\mathcal{F}(\lambda)$ that achieves correctness over $\mathcal{H} \subset \mathcal{F}(\lambda)$, the spirit of the security notion is to allow the adversary to also submit evaluation requests over circuits in $\mathcal{F}(\lambda) \setminus \mathcal{H}$ such that it eventually ends up observing incorrect decryptions. While this gap was exploited in several attack strategies against lattice-based FHE [30,31], CPA^D security is conceptually not difficult to achieve without requiring for correctness to hold over the whole set $\mathcal{F}(\lambda)$ (although this is not the point of this paper we illustrate this in appendix Sect. B.3 for self-containedness).

Application-aware Homomorphic Encryption. Following several CPA^D attacks against either "exact" and approximate schemes [30,31,44], Alexandru et al. [4,5] have proposed a new *weaker* variant of CPA^D security over restricted sets of circuits, termed Application-Aware HE security, that we denote AAHE to avoid confusion with the baseline notion. We provide more details in App. E.2. We emphasize that, as convenient as the AAHE model might be, full-blown CPA^D security is still quite interesting to investigate from a theoretical viewpoint and also of practical relevance (as many constructions manage to achieve it, e.g. [47,19,26,57], see also Sect. B.3). Additionally, we stress that, *while they are all genuine CPA^D attacks, none of the attacks that are presented in this paper go through the AAHE model.*

vCCA security (and others). As introduced in [52], given an FHE scheme $\mathcal{E}_H = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ the vCCA experiment is,

$$
\begin{aligned}
&(\mathsf{ek}, \mathsf{sk}) \leftarrow \mathcal{E}_H.\mathsf{KeyGen}(1^\lambda),\\
&(m_0, m_1, S) \leftarrow \mathcal{A}_1^{[\mathcal{E}_H.\mathsf{Enc}(.,\mathsf{ek})], \mathcal{E}_H.\mathsf{Dec}(.,\mathsf{sk})}([\mathsf{ek}], \mathsf{aux}),\\
&\gamma \xleftarrow{\$} \{0,1\},\\
&c^* \leftarrow \mathcal{E}_H.\mathsf{Enc}(m_\gamma, \mathsf{ek}),\\
&\gamma' \leftarrow \mathcal{A}_2^{[\mathcal{E}_H.\mathsf{Enc}(.,\mathsf{ek})], \mathcal{O}_{\text{vCCA}}(.,c^*,\mathsf{sk},\mathsf{aux})}(S, c^*, [\mathsf{ek}], \mathsf{aux}),\\
&\gamma = \gamma',
\end{aligned}
$$

with, letting $(f, c_0, ..., c_{L-1}) \leftarrow \mathsf{extract}(c, \mathsf{aux})$,

$$\mathcal{O}_{\text{vCCA}}(c, c^*, \mathsf{sk}, \mathsf{aux}) = \begin{cases} \mathcal{E}_H.\mathsf{Dec}(c, \mathsf{sk}) & \text{if } c^* \notin \{c_0, ..., c_{L-1}\}, \\ \perp & \text{otherwise.} \end{cases} \tag{6}$$

In the above, $\mathsf{extract} : \mathcal{C} \times \mathcal{X} \longrightarrow \mathcal{F}(\lambda) \times \mathcal{C}^*$, where $\mathcal{X}$ denotes a set of auxiliary data, is a PPT witness extractor[9]. In essence, the vCCA game is exactly the single challenge CCA2 game, with the second step decryption oracle being augmented in order to filter out *all* byproducts of the challenge ciphertext (rather than just the challenge ciphertext). In [52], vCCA security is defined and studied under the $\mathcal{F}(\lambda)$-correctness assumption for the underlying FHE scheme, but the notion still makes sense when that assumption is not satisfied [25].

We also consider a few additional CCA security notions for FHE, namely vCCA^D [25], (IND-)SA [63] and the more recent IV-CCA [65] security. We defer their definitions to Sect. 4.4 where they are needed. See also Fig. 1.

2.3 The basic Regev Scheme ($\mathcal{R}$)

In this paper, we use the well-known Regev scheme and its simple RLWE variant as a running example. For Regev, we first consider the symmetric variant, which is parameterized by a security parameter λ, a dimension n, an integer q and a (discrete Gaussian) probability distribution χ_{σ_0} on $\mathbb{Z}_q$ with standard deviation σ_0. Plaintexts are elements of $\mathbb{Z}_t$ and ciphertexts are elements of $\mathbb{Z}_q^n \times \mathbb{Z}_q$. The scheme $\mathcal{R}(\lambda, n, q, t, \sigma_0)$ is then defined as follows:

- $\mathcal{R}.\mathsf{KeyGen}$: pick a secret key $\mathsf{sk} \in \mathbb{Z}_q^n$ uniformly (or another appropriate dist.).
- $\mathcal{R}.\mathsf{Enc}$: given a plaintext $m \in \mathbb{Z}_t$, pick $a \in \mathbb{Z}_q^n$ uniformly at random, pick e in $\mathbb{Z}_q$ according to χ_{σ_0}, and return (a, b) with $b = \langle a, \mathsf{sk} \rangle + \Delta m + e$ and $\Delta = q/t$.
- $\mathcal{R}.\mathsf{Dec}$: given a ciphertext $c = (a, b) \in \mathbb{Z}_q^{n+1}$, return $\left[\left\lceil \frac{1}{\Delta}(b - \langle a, \mathsf{sk} \rangle) \right\rfloor\right]_t$.

This scheme is trivially linearly homomorphic, offering homomorphic addition, mult-by-plaintext ($\mathsf{MulByPlain}(\alpha, .)$, $\alpha \in \mathbb{Z}_t$) and add-plaintext ($\mathsf{AddPlain}$) operators. The scheme further admits a well-known asymmetric variant [59,60] in which the public-key is defined as a large enough set of encryptions of 0 and encryption is performed by summing a random subset of these ciphertexts together with a trivial noiseless encryption of the message, $(\vec{0}, \Delta m)$.

Regarding the security notions of interest to us in this paper, it is well-known that $\mathcal{R}$ is CPA secure under the LWE assumption [59]. However, $\mathcal{R}$ is CPA^D insecure [30,31] and also trivially CCA1 insecure. For example, the ciphertext $(-\Delta \mathbf{1}_i, 0)$ decrypts to $[\mathsf{sk}_i]_t$, where $\mathbf{1}_i$ is the i-th standard basis vector, which is

[9] The spirit of the vCCA security notion is (at least) to model construction blueprints which embed proof material in their ciphertexts and rely on a SNARK to enforce correct homomorphic evaluations over some input ciphertexts. In this context, the above $\mathsf{extract}$ thus corresponds to the extractor of that underlying SNARK which allows one to retrieve a witness from the proven statement as well as auxiliary data, see [52] for more details.

enough to recover the secret key with n decryption requests when its components fit in $\mathbb{Z}_t$. When, as above, sk is picked uniformly in $\mathbb{Z}_q$, it is also easy for a CCA1 adversary to retrieve sk by means of these kinds of decryption requests (we describe such an attack in App. D.1 for completeness).

The scheme also has an elementary ring variant, RLWE, working over $R_q = \mathbb{Z}_q[X]/(X^n+1)$ [50,51]:

- RLWE.Enc: given $m_0, \ldots, m_{n-1} \in \mathbb{Z}_t^n$ return (a, b) where $b = a \cdot \mathsf{sk} + \Delta m + e$, with $m(X) = \sum_{i=0}^{n-1} m_i X^i$.
- RLWE.Dec: given $(a, b) \in R_q^2$, return $\left[\left\lceil \frac{1}{\Delta}(b - a \cdot \mathsf{sk}) \right\rfloor\right]_t$ where the rounding is performed coefficient-wise.

The above can also be turned asymmetric by a number of well-known means. As Regev, RLWE is trivially CCA1 insecure, e.g. the ciphertext $(-\Delta \cdot 1, 0)$ decrypts to sk (at least for a binary sk, see also App. D.1).

It is also well-known that both Regev and its above RLWE variant can be rendered fully homomorphic by several means. *Lastly, we assume that the reader is familiar with the well-known fact that the mainstream FHE, namely BFV, BGV and TFHE, in their somewhat or levelled flavors, are all extensions of either Regev or the above RLWE variant*[10]*. This is also so for CKKS, with some subtleties that will be discussed when needed.*

2.4 Smudging

Smudging is a technique that consists in "hiding" a small noise by flooding it in a much larger noise such that the effect of the small noise becomes negligible. Smudging was first introduced in [10] in the context of threshold PKE and later, in the context of threshold FHE in [7,16] (essentially to make sure 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 proposals, e.g. [54]. Beyond threshold FHE, smudging has also been suggested as a countermeasure to CPA^D attacks against CKKS [48] (where it is referred to as noise flooding) or the other "exact" FHE [30]. Smudging comes into different flavors depending on whether the statistical distance or the Rényi divergence is considered [36,20,53] or whether worst-case/non-worst-case smudging should be performed [11,44,58,12]. In this work, for simplicity sake, we consider *worst-case* smudging based on the *statistical distance*. More specifically, we rely only on simple "Smudging Lemmas" such as the following [7, Lemma 1].

Lemma 1 (Smudging Lemma [7]). *Let B_0 and B_1 be two positive integers and let $e_0 \in [\![-B_0, B_0]\!]$ be a fixed integer. Let e_1 be chosen uniformly at random in $[\![-B_1, B_1]\!]$. Then, if $B_1 \geq 2^\lambda B_0$ the statistical distance between the distribution of e_1 and that of $e_0 + e_1$, $d(f_{e_1}, f_{e_1+e_0})$, is bounded by $negl(\lambda)$.*

The above Lemma is useful as it shows that a centered uniform noise with an appropriately large support can "smudge out" a constant value and, more

[10] See e.g. [30, Sect. 4] if needed.

generally, any random variable following a distribution with a bounded support. We can further extend it in order to "smudge out" a discrete Gaussian noise.

Lemma 2. *Let ε be a centered discrete Gaussian random variable with variance σ_0^2. Further let $B_0 = \sigma_0\sqrt{2(\lambda+1)\log 2}$, then $P(\varepsilon \notin [-B_0, B_0]) \leq 2^{-\lambda}$.*

Proof. Immediate from the Chernoff bound for the (centered) discrete Gaussian distribution (e.g. [28, Corollary 17]). □

If we now choose $B_1 = 2^\lambda B_0 = 2^\lambda \sigma_0\sqrt{2(\lambda+1)\log 2}$, then Lemma 1 applies, directly leading the following Lemma.

Lemma 3. *Let ε be a centered discrete Gaussian random variable with variance σ_0^2 and let $B_1 \geq 2^\lambda \sigma_0\sqrt{2(\lambda+1)\log 2}$, then $d(f_v, f_{v+\varepsilon}) \leq 2^{-\lambda}$, where v is picked uniformly in $[\![-B_1, B_1]\!]$.*

Although we will use the uniform distribution from Lemma 1 for smudging, other distributions, such as the discrete Gaussian [47,54] can be used. However, the results in this paper are only marginally impacted from such a distribution change.

3 HELLHO schemes: definitions and first properties

In this paper, we consider the special case of *at least* linearly homomorphic schemes for which the homomorphic addition operator is linear over the ciphertext domain. We refer to such schemes as HELLHO (for "Homomorphic Encryption schemes with Linear Linearly Homomorphic Operators").

3.1 HELLHO and $\mathbb{F}$-HELLHO schemes

Definition 4 (HELLHO scheme). *Let*

$$\mathcal{E}_H = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, [\mathsf{AddPlain}, \mathsf{Mul}, \mathsf{MulByPlain}, ...]),$$

denote an (at least) additively homomorphic encryption scheme over $\mathcal{F}(\lambda)$ with plaintext domain $(\mathcal{P}, +)$ and ciphertext domain $(\mathcal{C}, +)$ two finite abelian groups such that the homomorphic evaluation of additions $\mathcal{E}_H.\mathsf{Add} : \mathcal{C}^2 \longrightarrow \mathcal{C}$ satisfies

$$\mathcal{E}_H.\mathsf{Add}(\mathsf{ct}_1, \mathsf{ct}_2) = \mathsf{ct}_1 + \mathsf{ct}_2.$$

Let us further assume that there is an explicit and efficiently computable group isomorphism such that $\mathcal{C} \cong \prod_{i=0}^{n-1} \mathbb{Z}_{q_i}$ for a family of integers $(q_i)_{i\in[\![0,n-1]\!]}$ with $\prod_i q_i \leq 2^{O(poly(\lambda))}$.

Then, we say that $\mathcal{E}_H$ is a HELLHO scheme when:

1. *There exists a polynomial P such that $P(\lambda)$ uniformly sampled elements from $\mathcal{C}$ form a generating family of $\mathcal{C}$ seen as a $\mathbb{Z}$-module, denoted $\mathbb{M}(\mathcal{C})$, with non-negligible probability.*

2. *For any* $m \in \mathcal{P}$, $\mathcal{E}_H.\mathsf{Enc}(m)$ *is indistinguishable from uniform over* $\mathcal{C}$ *(ciphertext pseudorandomness).*
3. *Given any* $\mathsf{ct} \in \mathcal{C}$ *and a generating family* $(e_i)_{i \in [\![0,N-1]\!]}$ *of* $\mathcal{C}$*, one can efficiently find* $\alpha_0, ..., \alpha_{N-1} \in \mathbb{Z}^N$*, bounded by* $\mathrm{lcm}_i\, q_i$*, such that* $\mathsf{ct} = \sum_{i=0}^{N-1} \alpha_i \cdot e_i$.

Intuitively, the class of HELLHO schemes is the class of at least additively homomorphic schemes with an homomorphic addition operator that allows one to perform linear algebra over their full ciphertext domain. We also highlight that property 3 in the above definition *does not place any low norm constraint on the* α_i*'s* (which jeopardizes $\mathcal{F}(\lambda)$-correctness, as we shall see in Sect. 4.2). For instance, property 3 is satisfied when the latter coefficients can simply be obtained, given ct and the e_i's, by means of gaussian elimination. We further emphasize that the scalar multiplication $\mathbb{Z} \times \mathcal{C} \longrightarrow \mathcal{C}$ that is induced by the $\mathbb{Z}$-module structure is *not* the $\mathcal{E}_H.\mathsf{MulByPlain} : \mathcal{P} \times \mathcal{C} \longrightarrow \mathcal{C}$ operator (if it exists[11]).

For convenience, we also instantiate the above definition in the simpler setting when $(\mathcal{C}, +)$ is isomorphic to a vector space over a finite field $\mathbb{Z}_p$ for a prime p[12].

Definition 5 ($\mathbb{F}$-HELLHO). *Let* $p \leq 2^{O(poly(\lambda))}$ *be prime, and,*

$$\mathcal{E}_H = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, [\mathsf{AddPlain}, \mathsf{Mul}, \mathsf{MulByPlain}, ...]),$$

denote an (at least) additively homomorphic encryption scheme over $\mathcal{F}(\lambda)$ *with plaintext domain* $\mathcal{P}$ *and ciphertext domain* $\mathcal{C}$*, such that there is an explicit and efficiently computable group isomorphism between* $(\mathcal{C}, +)$ *and* $(\mathbb{Z}_p^N, +)$ *for* $N \leq O(poly(\lambda))$*. Then,* $\mathcal{E}_H$ *is an* $\mathbb{F}$*-HELLHO scheme when the homomorphic evaluation of additions,* $\mathcal{E}_H.\mathsf{Add} : \mathcal{C}^2 \longrightarrow \mathcal{C}$*, is such that,* $\mathcal{E}_H.\mathsf{Add}(\mathsf{ct}_1, \mathsf{ct}_2) = \mathsf{ct}_1 + \mathsf{ct}_2$*, and, for any* $m \in \mathcal{P}$*,* $\mathcal{E}_H.\mathsf{Enc}(m)$ *is indistinguishable from uniform over* $\mathcal{C}$ *(ciphertext pseudorandomness).*

Intuitively, the class of $\mathbb{F}$-HELLHO schemes is the class of at least additively homomorphic schemes such that their ciphertext domain together with their homomorphic addition operator define a *vector space* that can be efficiently identified to $\mathbb{Z}_p^N$. Hence, Properties 1 and 3 in Def. 4 are satisfied for $\mathbb{F}$-HELLHO's.

We now prove a number of simple lemmas on $\mathbb{F}$-HELLHO schemes which can also be easily generalized to full-blown HELLHO ones (or are implied by the properties explicited in the more general Def. 4). First, the next Lemma shows that Property 1 and Property 3 in Def. 4 follow implicitly from the vector space structure underlying the definition of $\mathbb{F}$-HELLHO schemes.

Lemma 6. *Let* $\mathcal{E}_H$ *be an* $\mathbb{F}$*-HELLHO scheme with ciphertext domain* $\mathcal{C} \cong \mathbb{Z}_p^N$*. Then, a set of* N *uniformly picked elements in* $\mathcal{C}$ *forms a basis of* $\mathbb{Z}_p^N$ *with non-negligible probability. Furthermore, given any* $\mathsf{ct} \in \mathcal{C}$ *and a basis,* $e_0, ..., e_{N-1}$*, for* $\mathbb{Z}_p^N$*, one can efficiently find* $\alpha_0, ..., \alpha_{N-1} \in \mathbb{Z}_p^N$ *such that* $\mathsf{ct} = \sum_{i=0}^{N-1} \alpha_i \cdot e_i$.

[11] E.g., when $\mathcal{P} = \mathbb{Z}_t$ and $\mathcal{C} = \mathbb{Z}_q^N$, $\mathcal{E}_H.\mathsf{MulByPlain} : \mathbb{Z}_t \times \mathbb{Z}_q^N \longrightarrow \mathbb{Z}_q^N$ (vs $\mathbb{Z}_q \times \mathbb{Z}_q^N \longrightarrow \mathbb{Z}_q^N$).

[12] Note that if $(\mathcal{C}, +)$ can easily be identified to $(\mathbb{F}_{p^k}^N, +)$, then it can easily be identified to $(\mathbb{Z}_p^{k \cdot N}, +)$ as an additive group. Thus, the $\mathbb{F}$-HELLHO definition holds for any finite field.

Proof. Consider a set of N uniformly picked vectors in $\mathbb{Z}_p^N$. The probability, μ, that they are linearly independent and thus form a basis of $\mathbb{Z}_p^N$ is[13],

$$\prod_{k=0}^{N-1}\left(1-\frac{1}{p^{N-k}}\right)\geq\left(1-\frac{1}{p}\right)^2$$

(see Lemma 20 in Appendix B.1, p. 34). Assume we want that $\mu\geq\frac{1}{4}$. Then $\left(1-\frac{1}{p}\right)^2\geq\frac{1}{4}$ yields $p\geq 2$ which is obviously true of any prime.

The second part of the statement then follows from the vector space structure of $\mathbb{Z}_p^N$: the α_i's can be found by means of Gaussian elimination with complexity in $O(N^3\log_2 p)$, i.e. $O(\text{poly}(\lambda))$ (from Def. 5). □

Lemma 7. *Let $\mathcal{E}_H$ be an $\mathbb{F}$-HELLHO scheme with ciphertext domain $\mathcal{C}\cong\mathbb{Z}_p^N$. For any $m\in\mathcal{P}^N$, the set $\mathsf{ct}_0,\ldots,\mathsf{ct}_{N-1}$ with $\mathsf{ct}_i=\mathcal{E}_H.\mathsf{Enc}(m_i)$, for $i\in[\![0,N-1]\!]$, forms a basis of $\mathbb{Z}_p^N$ with non-negligible probability. Also, $\mathcal{E}_H$ is IND-CPA secure.*

Proof. The first part of the statement follows from Lemma 6 above and the ciphertext pseudorandomness property of $\mathcal{E}_H$. The second part of the statement follows from the fact that ciphertext pseudorandomness trivially implies SIM-CPA security, the latter being equivalent to IND-CPA (see App. E.5). □

This Lemma carries over to the more general HELLHO setting (from Property 1 and Property 2 in Def. 4).

Lemma 8. *Let $\mathcal{E}_H$ be an $\mathbb{F}$-HELLHO scheme with ciphertext domain $\mathcal{C}\cong\mathbb{Z}_p^N$. Then the scalar multiplication over $\mathbb{Z}_p^N$ can be efficiently performed using only legitimate homomorphic operations from $\mathcal{E}_H$, even limited to $\mathcal{E}_H.\mathsf{Add}:\mathcal{C}^2\longrightarrow\mathcal{C}$.*

Proof. Denote $\mathcal{E}_H.\mathsf{Add}$ as $\oplus$. Then, given $\alpha\in\mathbb{Z}_p$ and $\mathsf{ct}\in\mathcal{C}$, the scalar multiplication over $\mathbb{Z}_p^N$ is implemented by $\mathsf{ScalMul}_p:\mathbb{Z}_p\times\mathcal{C}\longrightarrow\mathcal{C}$ defined as,

$$\mathsf{ScalMul}_p(\alpha,\mathsf{ct})=\bigoplus_{j:\alpha_j=1}\mathsf{ct}_j, \tag{7}$$

where the α_j's are the digits in the base 2 decomposition of α, $\mathsf{ct}_0=\mathsf{ct}$ and $\mathsf{ct}_j=\mathsf{ct}_{j-1}\oplus\mathsf{ct}_{j-1}\equiv\mathcal{E}_H.\mathsf{Add}(\mathsf{ct}_{j-1},\mathsf{ct}_{j-1})$ $(j>0)$, i.e. $\mathsf{ct}_j=2^j\cdot\mathsf{ct}$.

The efficiency claim follows from the fact that only $O(\log_2\alpha)\leq O(\text{poly}(\lambda))$ calls to $\mathcal{E}_H.\mathsf{Add}$ are required to evaluate Eq. (7). □

This Lemma trivially carries over to the general HELLHO setting of Def. 4, yielding an algorithm, $\mathsf{ScalMul}:\mathbb{Z}\times\mathcal{C}\longrightarrow\mathcal{C}$, that efficiently computes the scalar multiplication over $\mathbb{M}(\mathcal{C})$ by means only of $\mathcal{E}_H.\mathsf{Add}$, as well as to the case where $\mathcal{C}\cong\mathbb{Z}_q^N$ for a non prime modulus q.

Lemma 9. *Let $\mathcal{E}_H$ be an $\mathbb{F}$-HELLHO scheme with $\mathcal{C}\cong\mathbb{Z}_p^N$. Given any ciphertext $\mathsf{ct}\in\mathcal{C}$ and a basis, $e_0,\ldots,e_{N-1}$, of $\mathbb{Z}_p^N$, one can efficiently obtain ct from the e_i's by means only of legitimate homomorphic operations from $\mathcal{E}_H$.*

[13] Given k independent vectors, the next one must avoid their span S, with $|S|=p^k$.

Proof. From the vector space structure of $\mathbb{Z}_p^N$, given $\mathsf{ct} \in \mathcal{C}$, one can efficiently find $(\alpha_0, ..., \alpha_{N-1}) \in \mathbb{Z}_p^N$ such that

$$\mathsf{ct} = \sum_{i=0}^{N-1} \alpha_i \cdot e_i = \mathcal{E}_H.\mathsf{Add}(\mathsf{ScalMul}_p(\alpha_0, e_0), ..., \mathsf{ScalMul}_p(\alpha_{N-1}, e_{N-1})), \tag{8}$$

by means of Gaussian elimination, hence in $O(N^3 \log_2 p) \leq O(\mathrm{poly}(\lambda))$. From Lemma 8 above, it further follows that ct can be obtained from the e_i's and the α_i's by means only of legitimate homomorphic operations from $\mathcal{E}_H$. Still from Lemma 8, this requires $O(N \log_2 p) \leq O(\mathrm{poly}(\lambda))$ homomorphic additions. □

Again, this Lemma also carries over the more general HELLHO setting (as implied by Property 3 in Def. 4, rather than a vector space structure).

Overall, the above Lemmas thus mean that, given polynomially many fresh encryptions of 0, $e_0, ..., e_{N-1}$, and any $\mathsf{ct} \in \mathcal{C}$, an adversary can efficiently find a polynomial-size circuit, $C_{\mathsf{ct};(e_i)_{i\in[\![0,N-1]\!]}} \in \mathcal{F}(\lambda)$, such that $C_{\mathsf{ct};(e_i)_{i\in[\![0,N-1]\!]}}(0, ..., 0) = 0$ yet $\mathsf{Eval}(C_{\mathsf{ct};(e_i)_{i\in[\![0,N-1]\!]}}; e_0, ..., e_{N-1}) = \mathsf{ct}$. In the rest of this paper, $C_{\mathsf{ct};(e_i)_{i\in[\![0,N-1]\!]}}$ will always be specified by a linear combination with arbitrary large cofficients in $\mathbb{Z}_p$. *This trick will allow us to insert arbitrary ciphertexts into the internal state of a CPA^D challenger such that it sees them as encryptions of 0 and thus accepts decryption requests over them.* To illustrate this linear algebra-based ciphertext reconstruction technique we provide in Appendix F several numerical examples on the Regev scheme with toy parameters.

3.2 Concrete HELLHO schemes

The above definitions cover a relatively large class of schemes, starting from Regev (Sect. 2.3).

Proposition 10. *Let p be a prime, then $\mathcal{R}(\lambda, n, p, t, \sigma_0)$ is an $\mathbb{F}$-HELLHO scheme.*

Proof. Let $N = n + 1$. For that scheme, $\mathcal{P} = \mathbb{Z}_t$, $\mathcal{C} = \mathbb{Z}_p^N$ and $\mathcal{R}.\mathsf{Add}$ is the usual vector addition of $\mathbb{Z}_p^N$. Then, the scalar multiplication over $\mathbb{Z}_p^N$ can be implemented by $\mathsf{ScalMul}_p : \mathbb{Z}_p \times \mathcal{C} \longrightarrow \mathcal{C}$, defined as,

$$\mathsf{ScalMul}_p(\alpha, (a, b)) = ([\alpha a]_p, [\alpha b]_p),$$

which is equivalent to Eq. (7). Thus, $(\mathbb{Z}_p^N, +, \cdot)$, with both operators implemented respectively by means of $\mathcal{R}.\mathsf{Add}$ and $\mathsf{ScalMul}_p$, trivially has the expected $\mathbb{Z}_p$-vector space structure from an $\mathbb{F}$-HELLHO scheme.

Now, from the LWE assumption, for any $m \in \mathcal{P}^N$, the set $e_0, ..., e_{N-1}$, with $e_i = \mathcal{E}_H.\mathsf{Enc}(m_i)$ for $i \in [\![0, N-1]\!]$, is indistinguishable from a set of N uniformly picked vector in $\mathbb{Z}_p^N$. From Lemma 6, the probability that they are linearly independent and thus form a basis of $\mathbb{Z}_p^N$ is non-negligible. □

Remark that the above result also applies to the ring variant of Regev, RLWE, that we introduced in Sect. 2.3: even if for this scheme $\mathcal{C} = \mathbb{Z}_p[X]/(X^n + 1) \times$

$\mathbb{Z}_p[X]/(X^n+1)$ is given a polynomial ring structure, $\mathcal{C}$ interpreted as $\mathbb{Z}_p^{2n}$ yields the vector space structure that the $\mathbb{F}$-HELLHO scheme definition expects. Thus, the proof of the previous proposition works (with $N = 2n$).

The above Proposition further generalizes to the HELLHO, rather than $\mathbb{F}$-HELLHO, case where $\mathcal{C} = \mathbb{Z}_q^n$ with q not necessarily a prime power. This is so since Def. 5 requires that a generating family of $\mathcal{C}$ interpreted as a $\mathbb{Z}$-module can essentially be obtained from a set of ciphertexts of (known) polynomial-in-λ size and also makes explicit the existence of an efficient algorithm for finding linear combinations (with arbitrary large coefficients in $\mathbb{Z}_q$).

More generally, our HELLHO scheme definition encompasses the basic variants of many RLWE-based *levelled or somewhat* FHE, such as BFV [41], BGV [23] and CKKS [33], which are multiplicative extensions of the vanilla RLWE scheme of Sect. 2.3. Let us emphasize that bootstrapping does not prevent a scheme from belonging to the HELLHO class, *as long as the scheme's mode of operation allows unrestricted use of the natural homomorphic addition* (note that this is by far the most common mode of operation for the aforementioned RLWE-based schemes). Thus, extensions of Regev with bootstrapping such as TFHE [34,35] or FINAL [18] are also covered by the definition when their mode of operation allows for "unchecked addition" (i.e. homomorphic addition not immediately followed by bootstrapping[14]). Lastly, it can easily be verified from the definition of these schemes[15], that the HELLHO class also covers GSW [43], the more recent code-based somewhat homomorphic scheme in [2] and LIP-based FHE in [45,24], as well as the LPN-based scheme in [15] that was broken in [21] by jointly exploiting its linear decryption function as well as its ability to homomorphically evaluate the majority function. Thus, even linear-only homomorphic variants of this latter scheme, which evade [21], are covered by our results.

Conversely, HELLHO schemes are not meant as an abstraction of all FHE schemes and leave many of them uncovered, this is the case for all bootstrappable schemes (as many of the above) *as soon as inclusion of a bootstrapping operation is explicitly required in the mode of operation of the homomorphic addition operator* which then becomes non-linear over $\mathcal{C}$[16]. Also, the definition does not cover number-theoretic schemes such as Paillier, ElGamal or others as their homomorphic addition operators are also nonlinear over their ciphertext domain.

Still, the spirit of our definitions is to primarily account for the lattice-based levelled and somewhat FHE schemes for which CPA^D security issues are the most severe since they allow for uncontrolled noise growths. Although their vari-

[14] To the best of our knowledge, this is still the case in popular libraries such as TFHELib or TFHE-rs.

[15] In all these schemes the ciphertext domain is isomorphic to $\mathbb{Z}_q^N$, the homomorphic addition operator is the natural component-wise addition and their underlying security assumption implies ciphertext pseudo-randomness. For example, when q is prime, the proof of Prop. 10 thus also applies to these schemes.

[16] Recall that CPA^D security models "passive" adversaries which follows the cryptosystem specifications. So if the mode of operation of the homomorphic addition explicitly includes bootstrapping, then a CPA^D adversary will not do otherwise. Of course, CCA adversaries are not bound to such "kindness".

ants with systematic bootstrapping may be successfully attacked in the CPA^D setting for some parameters (for example [31] successfully attacks TFHE in gate-bootstrapping mode when the probability of bootstrapping error is non-negligible in the security parameter λ), they can be parameterized to achieve statistical $\mathcal{F}(\lambda)$-correctness [30, Sect. C] to solve CPA^D security issues (as CPA^D then downgrades to CPA). *We also highlight that when $\mathcal{F}(\lambda)$-correctness implies CPA^D security [46, Lemma 1], the converse is not true.*

Lastly, we emphasize that the HELLHO class also models cases where noise management is done within the ciphertexts themselves (as in the DE-RLWE and DE-CKKS schemes studied in Sect. 5.1). Also, it is interesting to note that error management is *not* inherent in building useful homomorphic schemes. For example, [55] presents a variant of GSW with trapdoor-based decryption, GG-GSW, that is plausibly CCA1-secure. While the precise assumptions under which this holds are still under discussion within the community [56], this construction is particularly relevant to our work: being a HELLHO scheme (the GSW ciphertext structure and addition operator are left unchanged), it may fill the hole we will leave open in Sect. 4.3 on the existence of CCA1 secure HELLHO schemes. It may therefore also be CPA$_1^D$-secure. Interestingly, GG-GSW neither performs any noise management nor achieves $\mathcal{F}(\lambda)$-correctness (although its parameters can be chosen such that it is useful, i.e. provides correct results with high probability for some circuits).

4 Impossibility results

4.1 On the existence of CPA^D-secure HELLHO

In this section, we establish that HELLHO schemes achieving CPA^D security do not exist. We do so by exhibiting a CPA^D attack that works *unconditionally* on any such scheme and, in particular, independently of any security assumption (contrary to all known attacks on (R)LWE-based schemes [46,47,44,30,31]).

Proposition 11. *Let $\mathcal{E}_H$ denote a (loosely correct) HELLHO scheme, then $\mathcal{E}_H$ is not CPA^D secure.*

Proof. This is proved by the existence of the following attack that operates in the CPA^D model. Indeed, a CPA^D adversary $\mathcal{A}$ may then proceed as follows:

- $\mathcal{A}$ requests encryptions of 0 from its challenger's encryption oracle until it gets a generating set, $e_0, ..., e_{N-1}$, of $\mathbb{M}(\mathcal{C})$ (from Property 1 and Property 2 in Def. 4). As a result, the CPA^D challenger's internal state contains the following N triplets (at indices $0, ..., N-1$),

$$[(0, 0, e_0), ..., (0, 0, e_{N-1})].$$

- $\mathcal{A}$ issues a challenge request with messages $m_0 = 0$ and $m_1 = 1$ to get ct^*, an encryption of m_γ. As a result, the CPA^D challenger's internal state further contains triplet $(0, 1, \mathsf{ct}^*)$ at index N.

– $\mathcal{A}$ then efficiently finds $\alpha_0^*, ..., \alpha_{N-1}^* \in \mathbb{Z}^N$ (from Property 3 in Def. 4) so that

$$\mathsf{ct}^* = \sum_{i=0}^{N-1} \alpha_i^* \cdot e_i = \mathcal{E}_H.\mathsf{Add}(\mathsf{ScalMul}(\alpha_0^*, e_0), ..., \mathsf{ScalMul}(\alpha_{N-1}^*, e_{N-1})). \quad (9)$$

Note that $\mathcal{A}$ does this on its own and that the evaluation oracle is *not* involved at this stage.

– Following Lemma 9 (p. 14), $\mathcal{A}$ invokes the evaluation oracle to implement Eq. (9), i.e. following Eq. (7) to replace each of the (*nonlegit*) $\mathsf{ScalMul}$ invocations in (9) by several (*legit*) invocations of $\mathcal{E}_H.\mathsf{Add}$, getting ct^* in return. Most importantly, this has *the effect of adding the triplet* $(0, 0, \mathsf{ct}^*)$ *in the internal state of the CPA^D challenger* at index $N+1$ (assuming wlog that only one evaluation request is needed). Remark that the left and right cleartext values in this latter triplet are 0 since the e_i's are all encryptions of 0.
– $\mathcal{A}$ then issues a decryption request on index $N+1$ towards its challenger which accepts it, since $S[N+1].m_0 = S[N+1].m_1 = 0$, and reveals $\mathcal{E}_H.\mathsf{Dec}(\mathsf{ct}^*)$.

Following our definition for loose correctness (Eq. 2, p. 6), the adversary then wins the game with probability at least $\frac{3}{4}$. □

Remark that the loose correctness requirement (which only applies to fresh ciphertexts) in the statement of the above proposition is sufficient. This is so, since the only ciphertext that $\mathcal{A}$ asks to decrypt is a perfect reconstruction (that ends up in the CPA^D challenger internal state along with 0 left and right cleartext values) of the challenge ciphertext, which is itself a fresh one. To help understanding the attack in the proof of the previous proposition and better picturing our linear algebra-based ciphertext reconstruction technique (and its effect on noise in this attack) we provide a numerical experiment on the Regev scheme, with toy parameters, in Appendix F.1.

The key intuition, when considering the Regev scheme as an example for this attack, is to understand that the addition of multiple carefully-chosen ciphertexts even with very large noises (for instance the intermediate ciphertexts obtained just before the last summation in Eq. 9) may produce another ciphertext that is interpreted as a ciphertext with small noise by the decryption function (in this case, the challenge ciphertext).

Keeping CKKS in mind, the above proof also applies to approximate schemes such that $\Pr_{r \in \mathsf{COIN}}(\|\mathsf{Dec}(\mathsf{Enc}(m; r)) - m\|_\infty \leq \varepsilon) \geq \frac{3}{4}$, for all $(\mathsf{ek}, \mathsf{sk}) \in \mathcal{K}$, all $m \in \mathcal{P}$ and a (small) fixed $\varepsilon \geq 0$ (recall also Eq. 4, p. 7) as long as $\|m_0 - m_1\|_\infty > 2\varepsilon$.

Also, remark that only one challenge ciphertext is required to perform the above attack. As a consequence, the previous proposition rules out both CPA^D security and also its strictly weaker single-challenge variant referred to as CPA$^D_2$ [25]. However, because the attack performs a post-challenge decryption request, the above result does not rule out CPA$^D_1$ security (as defined in Sect. 2.2). Since sCPA^D implies CPA^D, the previous result also rules out that HELLHO schemes can achieve this former stronger notion from [13] (similarly to the CPA^D case, we however do not rule out that sCPA$^D_1$ security, i.e. single challenge sCPA^D with the decryption oracle closing after challenge publication, may be achievable

by HELLHO schemes). Following Sect. 3.2, we emphasize that not belonging to the HELLHO class is necessary but not sufficient for CPA^D security.

4.2 On the existence of $\mathcal{F}(\lambda)$-correct HELLHO schemes

Proposition 12. *$\mathcal{F}(\lambda)$-correct HELLHO schemes do not exist.*

Proof. Recall Eqs. (1) and (3), p. 6. Let $\mathcal{E}_H$ denote a $\mathcal{F}(\lambda)$-correct (hence loosely correct) HELLHO scheme. From Lemma 7, $\mathcal{E}_H$ is CPA secure. Then, from Prop. 11, we also know that $\mathcal{E}_H$ is *not* CPA^D secure. However, from [46, Prop. 1], we additionally know that for $\mathcal{F}(\lambda)$-correct schemes, CPA^D security is equivalent to CPA security. The claim thus follows by contradiction. □

Prop. 12 also admits a more direct proof, which we also provide in appendix for completion (Prop. 21, p. 35).

4.3 Relation between CCA1 and CPA^D_1 for HELLHO schemes

We first establish the counter-intuitive fact that any CCA2 (or CCA1) attack against a HELLHO scheme can be turned into one that fits the CPA^D model.

Proposition 13. *Let $\mathcal{E}_H$ be a HELLHO scheme, for any efficient CCA2 adversary $\mathcal{A}$ against $\mathcal{E}_H$, there exists an efficient CPA^D adversary $\mathcal{B}$ against $\mathcal{E}_H$ that uses $\mathcal{A}$ as a subroutine and achieves the same advantage as $\mathcal{A}$.*

Proof. The reduction $\mathcal{B}$ proceeds as follows. First, $\mathcal{B}$ requests encryptions of 0 from the CPA^D challenger encryption oracle until it gets a generating set, $e_0, ..., e_{N-1}$, of $\mathbb{M}(\mathcal{C})$ (from Property 1 and Property 2 in Def. 4). As a result, the CPA^D challenger's internal state contains the N triplets,

$$[(0, 0, e_0), ..., (0, 0, e_{N-1})],$$

at indices $0, ..., N-1$. Then, $\mathcal{B}$ processes $\mathcal{A}$'s requests by doing the following:

- Encryption and challenge requests from $\mathcal{A}$ are transferred as is to the CPA^D challenger, with the reduction keeping track of the size of the challenger's internal state S.
- When $\mathcal{A}$ issues a decryption request over ciphertext $\mathsf{ct} \neq \mathsf{ct}^*$, then $\mathcal{B}$ efficiently finds on its own $\alpha_0, ..., \alpha_{N-1} \in \mathbb{Z}^N$ (from Property 3 in Def. 4) such that,

$$\mathsf{ct} = \sum_{i=0}^{N-1} \alpha_i \cdot e_i = \mathcal{E}_H.\mathsf{Add}(\mathsf{ScalMul}(\alpha_0, e_0), ..., \mathsf{ScalMul}(\alpha_{N-1}, e_{N-1})). \quad (10)$$

  Then, following Lemma 9, $\mathcal{B}$ invokes the evaluation oracle to implement Eq. (10), i.e. following Eq. (7) to replace the $\mathsf{ScalMul}$ invocations in (10), with the effect of adding the triplet $(0, 0, \mathsf{ct})$ in the internal state of the CPA^D challenger. Finally, $\mathcal{B}$ issues a decryption request with index $|S| - 1$ towards its CPA^D challenger (which accepts it since the left and right cleartext values in this latter triplet are both 0) and returns the result to $\mathcal{A}$.

□

Remark that we have *not* included a loose correctness requirement in the statement of the above proposition since the reduction in its proof works by reconstructing the very same ciphertexts that the CCA2 adversary passes as parameters to its decryption requests (so that they end up in the CPA^D challenger internal state along with 0 left and right cleartext values). Hence, a CCA2 adversary that works against a non-loosely correct scheme, yields a CPA^D adversary which works just as well against the same.

Corollary 14 (Prop. 13). *For the class of HELLHO schemes, CPA_1^D is equivalent to CCA1.*

Proof. CCA1 implies CPA_1^D [25]. In CCA1, all decryption requests are performed before the (first) challenge requests. Hence, the reduction in the proof of Prop. 13 works as is from a CCA1 adversary towards a CPA_1^D challenger, proving the other direction. □

This result is insightful since, as already discussed towards the end of Sect. 4.1, our CPA^D impossibility result (Prop. 11) does not rule out that CPA_1^D security may be achieved for HELLHO schemes. Furthermore, from a practical viewpoint, this result allows us to *turn any CCA1 attack against a HELLHO scheme into an equivalent CPA_1^D attack on the same scheme* by means of the reduction in the proof of Prop. 13. We will use this trick to build concrete CPA_1^D or even KR^D attacks on particular HELLHO schemes in Sect. 5.1. On the brighter side, Corr. 14 provides the key insight that proving CPA_1^D security for a HELLHO scheme is enough to establish its CCA1 security. Intuitively, the former security notion may be easier to establish than the latter. As a last insight, we highlight that since $\text{CCA1} \Rightarrow \text{sCPA}_1^D \Rightarrow \text{CPA}_1^D$, the above result also implies that sCPA_1^D is equivalent to CPA_1^D for HELLHO schemes.

To help understanding the reduction in the proof of Prop. 13 we provide in Appendix F.2 a numerical experiment turning a simple CCA1 attack against the Regev scheme, with toy parameters, into an equally successful CPA^D attack.

4.4 Impossibility of (IND-)SA, IV-CCA, vCCA and vCCA^D security for HELLHO schemes

We first start with an "easy pick" result that rules out both (IND-)SA [63] and vCCA^D [25] security for HELLHO schemes. vCCA^D security is a "CPA^D-style" generalization of vCCA and (IND-)SA security has been introduced to provide a meaningful security model for constructions ensuring integrity of FHE evaluations. We provide their definitions in App. E.3 and E.4 for completeness.

Corollary 15 (Prop. 11). *Let $\mathcal{E}_H$ denote a (loosely correct) HELLHO scheme, then $\mathcal{E}_H$ is neither vCCA^D nor (IND-)SA secure.*

Proof. The proof follows from the facts that both vCCA^D [25] and (IND-)SA (Prop. 24, App. E.4) security imply CPA^D security as well as from Prop. 11 which rules out CPA^D security for HELLHO schemes. □

Interestingly, the question of the existence of HELLHO schemes achieving vCCA security [52] remains so far unsettled. First, although vCCA was initially defined and studied under $\mathcal{F}(\lambda)$-correctness in [52], the notion still makes sense when that assumption is not true. In that regime, as established in [25], vCCA security is strictly weaker than vCCA^D security (even strictly weaker than the single challenge variant of the latter notion). Furthermore, it has also been shown in [25] that vCCA security only implies the strictly weaker "CCA1-style" variant of CPA^D, denoted CPA$_1^D$. As discussed towards the end of Sect. 4.1 (p. 18), however, Prop. 11, does not rule out CPA$_1^D$ security for HELLHO schemes. Hence, vCCA security for HELLHO schemes has to be ruled out by means of an argument not yet in our toolbox.

Proposition 16. *Let $\mathcal{E}_H$ denote a (loosely correct) HELLHO scheme, then $\mathcal{E}_H$ is not vCCA secure.*

Proof. First, $\mathcal{A}$ issues the challenge request with messages $m_0 = 0$ and $m_1 = 1$ towards the vCCA challenger to get ct^*, an encryption of m_γ. Then, $\mathcal{A}$ creates on its own enough well-formed encryptions of 0 until it obtains a generating set, $e_0, ..., e_{N-1}$, of $\mathbb{M}(\mathcal{C})$ (from Property 1 and Property 2 in Def. 4). It then efficiently computes $\alpha_0^*, ..., \alpha_{N-1}^* \in \mathbb{Z}^N$ (from Property 3 in Def. 4) such that

$$\mathsf{ct}^* = \sum_{i=0}^{N-1} \alpha_i^* \cdot e_i = \mathcal{E}_H.\mathsf{Add}(\mathsf{ScalMul}(\alpha_0^*, e_0), ..., \mathsf{ScalMul}(\alpha_{N-1}^*, e_{N-1})), \tag{11}$$

and, subsequently, produces a new ciphertext $\mathsf{ct}' = \mathsf{ct}^*$ by homomorphically evaluating Eq. (11) on its own following Lemma 9 as we have already done a number of times. Finally, $\mathcal{A}$ submits ct' to its vCCA challenger.

The key point that makes the attack work is as follows. First remark that, although $\mathsf{ct}' = \mathsf{ct}^*$, ct' has been produced via a legitimate homomorphic evaluation of a function $f^* \in \mathcal{F}(\lambda)$ that did not involve the challenge ciphertext as input (essentially, $\mathcal{A}$ has evaluated a challenge-dependent function over challenge-independent ciphertexts). Following Eq. (6), p. 10, and using the available auxiliary information, the vCCA challenger's decryption oracle extractor returns $(f^*, e_0, ..., e_{N-1})$ as a witness and the oracle thus accepts this decryption request since $\mathsf{ct}^* \notin \{e_0, ..., e_{N-1}\}$. From the loose correctness of $\mathcal{E}_H$ (recall Eq. 2, p. 6), this gives away $\mathcal{E}_H.\mathsf{Dec}(\mathsf{ct}') = \mathcal{E}_H.\mathsf{Dec}(\mathsf{ct}^*) = m_\gamma$ with probability at least $\frac{3}{4}$. □

The same proof strategy also applies to the weaker IV-CCA [65] security notion. In IV-CCA [65], $\mathcal{E}_H$ is augmented with an additional algorithm $\mathsf{Verif}_{\mathsf{pk}} : \mathcal{C} \times \mathcal{C}^L \to \{0, 1\}$. In the IV-CCA security game, the second step decryption oracle takes as input a vector CT of L ciphertexts and a ciphertext ct. It then proceeds as follows: if either $\mathcal{E}_H.\mathsf{Dec}(\mathsf{ct}) = \perp$ or $\mathcal{E}_H.\mathsf{Verif}_{\mathsf{pk}}(\mathsf{ct}, \mathsf{CT}) = 0$ or $\mathsf{ct}^* \in \mathsf{CT}$, then it returns $\perp$. Otherwise it returns $\mathcal{E}_H.\mathsf{Dec}(\mathsf{ct})$. In [65], the security notion is defined only under *perfect* $\mathcal{F}(\lambda)$-correctness, but, as for vCCA, it still makes sense when this assumption does not hold. It is shown in [65] that vCCA implies IV-CCA. However, similarly to the vCCA case, since the IV-CCA second step decryption oracle filters out *all* byproducts of the challenge ciphertext, IV-CCA implies only CPA$_1^D$ and does not imply CPA$_2^D$.

Corollary 17 (Prop. 16). *Let $\mathcal{E}_H$ denote a (loosely correct) HELLHO scheme, then $\mathcal{E}_H$ is not IV-CCA secure.*

Proof. Assume that $\mathcal{E}_H.\mathsf{Verif}$ is such that

$$P\left(\mathcal{E}_H.\mathsf{Verif}_{\mathsf{pk}}(\mathsf{ct},\mathsf{CT})=1 \;\middle|\; \begin{array}{l}(\mathsf{pk},\mathsf{sk})=\mathcal{E}_H.\mathsf{KeyGen}(1^\lambda),\\ \exists m_0,...,m_{L-1}\in\mathcal{P}^L:\mathsf{CT}_i=\mathcal{E}_H.\mathsf{Enc}(m_i;\mathsf{pk}),\\ \exists f\in\mathcal{F}(\lambda):\mathsf{ct}=\mathcal{E}_H.\mathsf{Eval}(f;\mathsf{CT}_0,...,\mathsf{CT}_{L-1})\end{array}\right)>\mathrm{negl}(\lambda),$$

which is a very mild assumption if $\mathcal{E}_H$ is of any use. Then, recalling the notations in the proof of Prop. 16, we have that $\mathcal{E}_H.\mathsf{Verif}_{\mathsf{pk}}(\mathsf{ct}',e_0,...,e_{N-1})=1$ with non-negligible probability. It follows that the IV-CCA second step decryption oracle accepts a decryption request over $\mathsf{ct}',e_0,...,e_{N-1}$ and, hence, reveals $\mathcal{E}_H.\mathsf{Dec}(\mathsf{ct}')=\mathcal{E}_H.\mathsf{Dec}(\mathsf{ct}^*)=m_\gamma$ to $\mathcal{A}$ with non-negligible probability. □

The notion of IV-CCA security has been recently introduced in [65], where it is shown that *non-compact* IV-CCA secure schemes can be built in the standard model from the LWE assumption and the additional assumption that *perfectly* $\mathcal{F}(\lambda)$-correct FHE can be built from LWE.This result does not contradict Corr. 17 as the resulting scheme is not a HELLHO one. First and foremost, because no HELLHO scheme can achieve the perfect $\mathcal{F}(\lambda)$-correctness requirement of the construction (from Prop. 12). Also, the construction blueprint in [65] implicitly requires that bootstrapping is applied as part of every homomorphic operation and the resulting (perfectly $\mathcal{F}(\lambda)$-correct) scheme is augmented with a proof system (in this case a non-interactive batch argument – BARG) for building Verif. Both aspects introduce nonlinearities in the scheme's homomorphic operators.

To conclude this section, we however leave open the question as to whether HELLHO schemes may achieve CCA1 security. Indeed, all the attacks in this section perform (one) post-challenge decryption requests, and hence do not fit the CCA1 model. We further highlight that the recent CCA1 and PCA1 impossibility results in [61,62] do not help to settle this question, as they apply to a class of *perfectly correct* schemes[17] which then excludes HELLHO ones (following Prop. 12). Still, for HELLHO schemes, we have shown that CCA1 is equivalent to CPA_1^D. Consistently, a close look at all known CPA^D attacks on CKKS, BFV, BGV and TFHE [46,30,31] tells that they fit the CPA_1^D (even KR^D) model.

5 Consequences for (R)LWE-based schemes with Dynamic Error Estimation (DEE)

In the context of CKKS, [47, Sect. 5] introduces and studies a category of schemes that use a ciphertext-dependent[18] noise variance estimation procedure to implement smudging in their decryption function. In a nutshell, that paper presents

[17] The notion of SHPKE considered in both papers, see e.g. [62, Sect. 3.1 and 3.2], intends to model schemes such as ElGamal and Paillier.

[18] Contrary to the more popular ciphertext-independent variance monitoring approach, which *may* yield CPA^D secure (non HELLHO) schemes. See App. B.3, p. 38.

an approach (which the authors of that paper attribute to Y. Polyakov) leveraging on a special message encoding which fixes many of the coordinates of CKKS message space to be 0 and to use these to obtain, at decryption, an estimation of the noise variance for the ciphertext. This estimation is then used to set the variance of their noise flooding mechanism (which essentially is equivalent to the noise smudging mechanism initially introduced in [7] for threshold FHE, Sect. 2.4). Then, still in the context of CKKS, the authors of [47] present CPA^D attacks on a "natural" class of such schemes that leverage on the noise/message dependencies which naturally arise in CKKS (and other schemes such as BGV and BFV) when performing homomorphic multiplications. However, this attack strategy leaves open the possibility that other scheme designs with dynamic error estimation may achieve CPA^D security or that weaker variants of CPA^D security (e.g. KR^D or non-adaptive CPA^D) might be achievable by such schemes. We now argue that the impossibility result given by Prop. 11 significantly extends the class of schemes for which dynamic error estimation techniques fail to provide CPA^D security. Further note that the attacks in [47] "only" exhibit some noise leakage on the norm of the cleartext inputs of a homomorphic computation and leave open the question "Can the leakage be *provably* limited to this information?" [47, Sect. 6]. Additionally, [47] explicitly leaves open the question of the KR^D security of schemes embedding dynamic error estimation. The attacks in the next sections provide negative answers to these open questions.

5.1 CPA^D security and the DEE heuristic

We now formally define the notion of an FHE scheme with DEE, starting from a vanilla (symmetric) RLWE scheme for simplicity-sake. The definition, however, straightforwardly carries over to other RLWE-based schemes such as BFV or BGV (or even CKKS), or schemes based on multi-secret LWE, as well as to the public-key settings for such schemes.

So let us consider the usual elementary ring variant of Regev, RLWE, working over $R_q = \mathbb{Z}_q[X]/(X^n+1)$ that we introduced in Sect. 2.3:

- $\mathsf{RLWE.Enc}$: given $m_0, \ldots, m_{n-1} \in \mathbb{Z}_t^n$ return (a, b) where $b = a \cdot \mathsf{sk} + \Delta m + e$, with $m(X) = \sum_{i=0}^{n-1} m_i X^i$.
- $\mathsf{RLWE.Dec}$ given $(a, b) \in R_q^2$, return $\left[\left\lceil \frac{1}{\Delta}(b - a \cdot \mathsf{sk}) \right\rfloor\right]_t$ where the rounding is performed coefficient-wise.

We leave unspecified the multiplication operator, as we do not need it (contrary to [47]) to carry out our attacks. In the case of CKKS, the decryption function is just $b - a \cdot \mathsf{sk}$. Note that, for simplicity-sake, *we set aside the CKKS message encoding, we sketch in App. C.3 how this restriction may be lifted.*

To turn RLWE into a scheme with DEE, $\mathsf{DE\text{-}RLWE}$, we then proceed by first reducing the plaintext domain of RLWE from $\mathbb{Z}_t^n$ to $\mathbb{Z}_t^k$ for some $0 < k < n$ (k is then the number of *payload slots* and $n - k$ the number of *noise slots*). Then,

- $\mathsf{DE\text{-}RLWE.Enc}$: given $m_0, \ldots, m_{k-1} \in \mathbb{Z}_t^k$, return

$$(a, b) = \mathsf{RLWE.Enc}(m_0, ..., m_{k-1}, \underbrace{0, ..., 0}_{n-k}).$$

– DE-RLWE.Dec: given $(a, b) \in R_q^2$, first compute $\mu = b - a \cdot \mathsf{sk}$. Then compute,

$$\hat{\sigma}^2 = \frac{1}{n-k} \sum_{i=k}^{n-1} \mu_i^2 \tag{12}$$

and finally return DE-RLWE.Smudge$(\mu_0, ..., \mu_{k-1}; \hat{\sigma}^2)$.

– DE-RLWE.Smudge: given $\mu_0, ..., \mu_{k-1} \in \mathbb{Z}_q^k$ as well as $\hat{\sigma}^2 \in \mathbb{R}^+$ first proceeds by picking a smudging noise vector v uniformly in $[\![-B_1, B_1]\!]^k$ with $B_1 = 2^\lambda u(\hat{\sigma})\sqrt{2(\lambda+1)\log 2}$ (recall Lemma 3, p. 12) with $u(\hat{\sigma}) \geq \hat{\sigma}$. Then several options are possible:
  - (Option 1) Return $\perp$ when $\|v\|_\infty \geq \frac{\Delta}{2}$ or, else, $[\lceil \mu_i/\Delta \rfloor]_t$, $i \in [\![0, k-1]\!]$.
  - (Option 2) Return $[\lceil (\mu_i + v_i)/\Delta \rfloor]_t$, $i \in [\![0, k-1]\!]$.
  - (Option 2/DE-CKKS variant [47, Sect. 5.1]) Return $\mu_i + v_i$, $i \in [\![0, k-1]\!]$.

The above scheme can easily be parameterized to achieve loose correctness[19] (recall Eq. 2, p. 6). *Additionally, DE-RLWE's homomorphic operators are the same as those of* RLWE.

The above blueprint, *which we now refer to as the DEE heuristic*, is appealing since CPA^D security can be easily achieved by means of smudging given an oracle that, for a ciphertext under an LWE or RLWE-based scheme, outputs the true variance of the noise in that ciphertext or an upper bound for it (App. B.3). The intuition is that with such an oracle, a reduction towards the CPA security of the scheme can answer decryption requests without actually decrypting (see Prop. 22 in App. B.3). In the above blueprint, the estimated variance $\hat{\sigma}^2$ is thus heuristically used in place of the true one. As appealing as it may seem, we now show that this heuristic approach cannot achieve CPA^D security.

Corollary 18 (Prop. 11). *Let $\mathcal{E}_H$ denote a (loosely correct) DE-RLWE scheme, then $\mathcal{E}_H$ is not CPA^D secure.*

Proof. RLWE is a HELLHO scheme and DE-RLWE is also so, as the DEE blueprint above modifies only the decryption function. The claim thus follows from the fact that CPA^D secure HELLHO schemes do not exist (Prop. 11). □

Furthermore, the attack in the proof of Prop. 11 also directly applies to the DE-CKKS scheme in [47, Sect. 5.1] which is a direct instantiation of the DEE heuristic (with Option 2 above for Smudge) with CKKS as the baseline RLWE scheme[20]. Although this scheme was already proven CPA^D insecure in [47, Theorem 6], we provide here a very different line of attack on this specific scheme.

An intuitive reason that may explain why the DEE heuristic fails is that for the variance estimation in Eq. (12) to make sense on a ciphertext, the noises

[19] Let γ_{n-k} be such that $P(\hat{\sigma} \leq \gamma_{n-k}\sigma_0 | \chi_{\sigma_0}) \geq \frac{3}{4}$, then choosing q s. t. $\frac{q}{2t} \geq \bar{B}_1 + 1$ with $\bar{B}_1 = 2^\lambda \gamma_{n-k}\sigma_0\sqrt{2(\lambda+1)\log 2}$ yields a loosely correct scheme.

[20] This scheme was initially implemented in the Palisade library and is presently supported in OpenFHE [8] where it corresponds to the mode where `CKKSDataType` is set to `REAL`. See `src/pke/examples/README.md` in [1] (`2edba9e`).

in the noise slots have to be iid. As soon as an adversary homomorphically computes ciphertext-dependent functions (which is the case of the linear combinations that we evaluate in our attacks), this iid assumption is jeopardized. This assumption can however be maintained in the *non-adaptive* CPA^D setting (where the adversary specifies all its requests right after seeing the public material of the scheme, Sect. 2.2). For example, in that latter restrictive setting which it refers to as CPA_0^D, [29, Sect. 4.2] provided a secure construction for a linearly homomorphic multi-secret LWE scheme instantiation of the DEE blueprint.

5.2 KR^D (in)security of DE-RLWE

Recall that our results for general HELLHO schemes do not rule out that such schemes may achieve CPA_1^D security, although Corr. 14 establishes that for this class of schemes CPA_1^D security is equivalent to CCA1 security. Furthermore, the proof of Corr. 14 establishes that any CCA1 attack against a HELLHO scheme can be turned into an equivalent CPA_1^D attack. We now use this fact to show that schemes from the DE-RLWE family cannot even achieve the much weaker notion of KR^D security (recall that KR^D<CPA_1^D<CCA1, Sect. 2.2 and E.1).

Proposition 19. *Let $\mathcal{E}_H$ be a DE-RLWE scheme, then $\mathcal{E}_H$ is not KR^D secure.*

Proof. To establish the claim, we show that a variant of the CCA1 attack in App. D.2 allows an adversary to retrieve the secret key of a DE-RLWE scheme. We then use Corr. 14 to turn this attack into a KR^D one.

We describe the CCA1 base for the attack considering *Option 1* for DE-RLWE's decryption function. The attack starts from a fresh well-formed encryption of 0, $c_0 = (a, b)$ with $b = a \cdot \mathsf{sk} + e$ under DE-RLWE. The adversary targets the first payload slot and starts a dichotomic search between 0 and Δ, guided by DE-RLWE.Dec, to find the critical value $\alpha^* \in \mathbb{Z}_q$ such that ciphertext $(a, b_0 + \alpha^*, b_1, ..., b_{k-1}, b_k, ..., b_{n-1})$ does not decrypt to 0 anymore while ciphertext $(a, b_0 + \alpha^* - 1, b_1, ..., b_{k-1}, b_k, ..., b_{n-1})$ still does. Remark that because the adversary does not modify the k last noise slots in c_0, the decryption function applies a smudging noise v based on (an estimation of) the fresh ciphertext variance which yields $\|v\|_\infty < \Delta/2$ with high probability (as high as the probability that a fresh well-formed ciphertext decrypts correctly) and therefore does not influence the dichotomic search. The adversary can then conclude that the noise e_0 in the first payload slot of c_0 is such that $e_0 + \alpha^* = \Delta/2$, i.e. that $e_0 = \Delta/2 - \alpha^*$, and thus gets one linear equation in the secret key sk. By repeating this process over n encryptions of 0[21], the adversary eventually retrieves sk by means of linear algebra. Remark also that this attack is clearly CCA1 as it does not use the legitimate add-plaintext operator which only allows one to add multiples of Δ to the payload slots in the b-term of an RLWE pair.

The claim then follows from the reduction in the proof of Prop. 13 which allows one to turn the above CCA1 attack into a CPA_1^D one. Finally, remark

[21] The attack can clearly be optimized to use only one encryption of 0, but this is not our point here.

that the resulting attack achieves a full key recovery and does not involve any challenge ciphertext, so it fits the KR^D setting. □

To achieve a full key recovery, the CCA1 attack needs to perform n dichotomic searches, each requiring the decryption of $O(\log_2 q)$ ciphertexts. Following the reduction in the proof of Prop. 13, the KR^D adversary needs to perform $O(n \log_2 q)$ calls to DE-RLWE.Add (to implement Eq. 10, p. 19) per such ciphertexts. Hence, assuming n and q respectively are in $O(\lambda)$ and $O(2^\lambda)$, the attack runs in $O(\mathrm{poly}(\lambda))$[22]. See also App. C.1 for a more refined analysis.

Note that the attack in the previous proposition works only against Option 1 in DE-RLWE's decryption function. Hence, the KR^D security of DE-RLWE (Option 2) and the DE-CKKS scheme of [47] is still not ruled out. We address this question for DE-CKKS, which was left as an open problem in [47], just next[23].

5.3 KR^D (in)security of DE-CKKS

In this section, we describe a concrete KR^D attack that practically works against DE-CKKS. In the spirit of the proof of Prop. 19, we do so by exhibiting a CCA1 key recovery attack against that scheme and then use the reduction in Prop. 13 to directly turn this attack into a KR^D one.

Outline of the attack. For simplicity sake, let us consider that the adversary attacks the first noise slot, i.e. of index k. Then, consider an encryption of 0, $\mathsf{ct} = (a, \underbrace{a \cdot s + e}_{b})$ under DE-CKKS. When this ciphertext is decrypted, we get $\hat{\sigma}^2 = \frac{1}{n-k} \sum_{i=k}^{n-1} e_i^2$. The smudging noise υ is then picked uniformly in $[\![-B_1, B_1]\!]^k$, with $B_1 = 2^\lambda \hat{\sigma} \sqrt{2(\lambda+1)\log 2}$. Hence, the decryption function returns a sample of k independent values, $\mu_0, \ldots, \mu_{k-1}$, of the form $\mu_i = e_i + \upsilon_i$, such that with overwhelming probability (from Lemma 2, p. 12),

$$|\mu_i| \leq B_0 + B_1 = \underbrace{\left(\sigma_0 + 2^\lambda \sqrt{\frac{1}{n-k} \sum_{i=k}^{n-1} e_i^2}\right) \sqrt{2(\lambda+1)\log 2}}_{u_{\text{payload}}}, \tag{13}$$

Remark that the adversary may arbitrarily increase the sample size by asking several decryptions of the same ciphertext[24].Additionally, for large enough k

[22] If q is a prime power, each of the $O(n \log_2 q)$ decryptions requires solving a linear system over $\mathbb{Z}_q^{2n}$ (in $O(n^3 \log_2 q)$) to find the coefficients of the linear combination allowing to rebuild the ciphertext (following Eq. 10) and additionally performing $O(n \log_2 q)$ homomorphic additions (each in $O(n \log_2 q)$). With n and q respectively in $O(\lambda)$ and $O(2^\lambda)$, this yields a complexity of $O(\lambda^6)$ for the adversary.

[23] We conjecture that a similar kind of attack works against DE-RLWE with Option 2 in its decryption function.

[24] This is so since the decryption function picks a fresh smudging noise each time it is called (so independently of the ciphertext noise, without relying on a random oracle).

(equivalently, n), we have,

$$\hat{u}_{\text{payload}} = \max_{i \in [\![0,k-1]\!]} |\mu_i| \approx u_{\text{payload}}. \tag{14}$$

We now consider a CCA1 adversary which tampers with the first noise slot by simply adding a constant integer value α, which may be negative (modulo q), to that slot, i.e. produces ciphertext,

$$(a, b_0, , ..., b_{k-1}, b_k + \alpha, b_{k+1}, ..., b_{n-1}), \tag{15}$$

and asks for its decryption to get a sample of k values bounded with overwhelming probability by,

$$u_\alpha = \left(\sigma_0 + 2^\lambda \sqrt{\frac{1}{n-k} \left((e_k + \alpha)^2 + \sum_{i=k+1}^{n-1} e_i^2 \right)} \right) \sqrt{2(\lambda+1)\log 2}, \tag{16}$$

instead of Eq. (13). Then, the adversary looks for α^* such that u_{α^*} is minimum by iterating over $[\![-B_0, B_0]\!]$ (recall again Lemma 2, p. 12). Since that minimum is obtained for $e_k + \alpha^* = 0$, i.e. $\alpha^* = -e_k$. The adversary then gains knowledge of e_k and therefore one linear equation in (a cyclic rotation of) sk. In the concrete attack, the adversary heuristically uses $\hat{u}_\alpha \approx u_\alpha$ (Eq. 14) to find the minimum by means of a basic degree-2 polynomial regression (see App. C.2 for completion).

To proceed with a full attack, the adversary may apply the same procedure to either or both the other noise slots in the same ciphertext or other ciphertexts until it gets n linear equations in sk and then may recover it by means of linear algebra. As already emphasized, because the ciphertext in Eq. (15) is not obtained by means of legitimate homomorphic operations, the above attack fits the CCA1 model. However, as already emphasized, Corr. 14 and in particular the reduction in the proof of Prop. 13 allow to turn any CCA1 attack against a HELLHO scheme into an equivalently successful CPA_1^D attack For clarity, we illustrate this mechanics by a numerical experiment with toy parameters in Appendix F.3. Furthermore, the resulting attack achieves a full key recovery and does not involve any challenge ciphertext, so it eventually fits the KR^D setting.

Besides, to turn the above CCA1 attack into the KR^D setting, we can do better than relying on the generic reduction in the proof of Prop. 13. Indeed, the KR^D adversary can solve a *single* linear system over $\mathbb{Z}_q^{2n}$ to generate ciphertext $(a; b) = (0, ..., 0; 0, ..., 0, 1, 0..., 0)$, i.e. with a 1 at index k in b, by means of Lemma 9. It then uses it to iterate over $\alpha \in [\![-B_0, B_0]\!]$ by means of calls to $\mathsf{DE\text{-}CKKS.Add}$, starting from ciphertext, $\mathsf{DE\text{-}CKKS.Add}(c_0, (0, ..., 0; 0, ..., 0, [-B_0]_q, 0))$, where ciphertext $(0, ..., 0; 0, ..., 0, [-B_0]_q, 0)$ is built from the former by means of Lemma 8. As a result, this KR^D variant of the attack needs only to perform two linear system resolutions, including the final one for recovering sk. For completeness, we provide more details in App. C.1.

Experimental results. Because the noise recovery part in the previous attack is heuristic, due to the estimation in Eq. (14), we have implemented it to show that it practically works. We report our results in Table 1. For these results,

n	$\log_2 q$	$n-k$	# ctxt	# samp	# dec/ctxt	# dec	UT (s)	FT (h)	20 threads
$\mathbf{2^{13}}$	**218**	**4096**	2	1	352256	704512	0.4263	0.959	6.4 m
$\mathbf{2^{14}}$	**438**	**8192**	2	1	704512	1409024	0.8798	4.004	25.7 m
$\mathbf{2^{15}}$	**767**	**16384**	2	1	1409024	2818048	1.7538	15.964	2.166 h
$\mathbf{2^{16}}$	**1553**	**32768**	2	1	2818048	5636096	3.6539	66.517	5.956 h

Table 1. Experimental results for the noise recovery part of the KR^D attack of Sect. 5.3 against DE-CKKS. See Sect. "Experimental results" on p. 27 for further details.

the "standard" $\sigma_0 = 3.19$ has been used, the parameters all achieve around 128 bits of security or more (the "small", for $n = 8192$, parameter set comes from [32] and the larger ones are "standard" [6]). Then, n and $\log_2 q$ are the dimension and bit size of the ciphertext modulus, and $n-k$ indicates the number of noise slots (over the n slots). We emphasize that OpenFHE's `REAL` CKKS mode, as well as [47], explicitly requires that $k = n-k = n/2$ (with n a power of 2), so we followed this requirement in Table 1. Column "# ctxt" is the number of encryptions of 0 required to carry out the attack (assuming the adversary exploits all the noise slots in a given ciphertext, so this is equal to $\lceil \frac{n}{n-k} \rceil$). Column "# dec/ctxt" indicates the number of times the adversary invokes the decryption oracle on such a ciphertext. This latter number corresponds to $2B_0$ (recall again Lemma 2, p. 12, for $\sigma_0 = 3.19$ and $\lambda = 128$, $B_0 = 43$) times the number of noise slots, $n-k$, times the number of decryptions ("# samp") the adversary has to do to extend the size of the sample over which it computes $\hat{u}_{\text{payload}}$ (as each decryption samples a fresh smudging noise). Then, "# dec" provides the number of times the decryption oracle is called to carry the complete attack. In all our experiments, we observed n correct noise guesses in a row, thus leading a successful key recovery. To check that the attack succeeded, the adversary can simply try to decrypt a few additional encryptions of 0 with the candidate key, getting low L_2-norm results when that candidate key is the correct one. Column "UT" gives the time required for one noise extraction using a non-optimized proof-of-concept Python implementation running on an average laptop PC. Lastly, column "FT" illustratively gives the time required for n noise extractions (yielding a full key recovery) in the same conditions (so it is representative of the time of the "CCA1 core" of the KR^D attack). Since each noise extraction is independent of the others, the attack is highly parallelizable. Thus, the last column provides additional runtimes on a highend laptop with 14 physical cores accounting for 20 HW threads.

Variance-based variant. The attack in this section has a natural variance-based variant in which Eq. (13), (14) and (16) are respectively replaced by

$$\sigma^2_{\text{payload}} = \sigma_0^2 + \frac{(2B_1+1)^2-1}{12} = \sigma_0^2 + \frac{(2^{\lambda+1}\sqrt{2\hat{\sigma}^2(\lambda+1)\log 2}+1)^2-1}{12},$$

as well as, $\hat{\sigma}^2_{\text{payload}} = \frac{1}{k}\sum_{i=0}^{k-1}\mu_i^2 \approx \sigma^2_{\text{payload}}$, and, finally,

$$\sigma_\alpha^2 = \sigma_0^2 + \frac{2^{\lambda+1}\sqrt{\frac{2}{n-k}\left((e_k+\alpha)^2+\sum_{i=k+1}^{n-1}e_i^2\right)(\lambda+1)\log 2}+1)^2-1}{12}.$$

Then $\hat{\sigma}_\alpha^2 \approx \sigma_\alpha^2$ is heuristically used to determine α^* such that $\sigma_{\alpha^*}^2$ is minimum. On experimental ground, the resulting noise estimation procedure is a bit less precise, leading to slightly less efficient attacks. Also, note that this variance-based variant of the attack can be modified to work when the smudging noise is picked from a (discrete) Gaussian distribution, as was explicitly done in [47]. We have also implemented the attack in that setting and observed that switching to a Gaussian smudging noise only marginally impacts the efficiency of the attack.

6 Beyond HELLHO schemes?

As a last remark, we would like to emphasize that the results in this paper are also applicable to construction blueprints in which a CPA^D secure FHE is augmented with the machinery required for proving that fresh and evaluated ciphertexts are well-formed (e.g. Encrypt-then-Sign or Encrypt-then-Prove in [52,25]) in an attempt to achieve some degree of CCA security. Indeed, although the augmented scheme may not be a HELLHO one, the intuition behind such construction strategies is that the proof machinery downgrades attackers to CPA^D ones onto the inner FHE scheme, which, if a HELLHO one, cannot achieve it.

7 Responsible disclosure

Although we did not test the attack in Sect. 5.3 directly against OpenFHE implementation of DE-CKKS, we disclosed a preliminary version of this paper to Yuriy Polyakov from the OpenFHE Team. This was done at the beginning of Jan. 2026. First, we all agreed that, although it is a valid CPA^D attack, our attack does not go through the AAHE security model (see [4] and also Sect. 2.2) under which OpenFHE implicitly operates. Then, we also agreed that, in DE-CKKS, where the noise stored in noise slots is seen only during the final decryption, intermediate noise overflows and large-noise ciphertexts cannot be detected. Finally, because our attack relies heavily on the fact that, as a result, we may encounter situations where the addition of multiple large-noise ciphertexts produces another ciphertext with small errors in the noise slots (recall the example in Sect. F.3), we all agreed that (at least) the options below are possible to mitigate the attack:

1. Add an HELib-style (static) estimator that updates a noise estimate for each operation. This additional (coarser) noise estimator could then detect overflows/large-noise components in addition to the (more precise) dynamic noise estimation within the DE-CKKS ciphertexts. Interestingly, the additional noise tracking would become part of the scheme specification and would then introduce some degree of non-linearity in the addition operator (see the discussion in App. B.3 on p. 38) that would make the extended scheme fall out of the HELLHO class and thus evade our impossibility result.
2. Switch from CPA^D security to AAHE security with a suitable restriction on the class of allowed circuits. This is a natural "countermeasure" as OpenFHE implicitly operates in this latter model.

3. Decrypt (selected) intermediate ciphertexts and record the dynamically estimated noises. Although this option probably makes less practical sense, it is at least theoretically possible and may be relevant in certain client-aided protocols related to funcCPA security [3] and others related notions [42].

Acknowledgments

The author would like to express warm thanks to Damien Stehlé who kindly reviewed a preliminary version of this work and made several insightful comments that lead to improvements. We also express our gratitude to Yuriy Polyakov for an in-depth technical discussion on the attack in Sect. 5.3 on DE-CKKS during Responsible Disclosure (the outcome of this discussion is summarized in Sect. 7) as well as for the several relevant suggestions he made to improve the paper. We thank David Pointcheval, Jérôme Nguyen and Michael Walter for also providing insightful feedback on our results. Lastly, we thank our CEA colleagues, Adda-Akram Bendouka, Aymen Boudguiga, Jean-Paul Bultel, Antoine Choffrut and Jules Dumezy who read early drafts and made suggestions. Our CEA colleague Alexandre Berne should also be credited for writing the parallel version of our attack on DE-CKKS.

References

1. OpenFHE. https://github.com/openfheorg/openfhe-development, [Online; accessed 18-Sept-2025]
2. Aguilar-Melchor, C., Dyseryn, V., Gaborit, P.: Somewhat homomorphic encryption based on random codes. Designs, Codes and Cryptography **93**, 1645–1669 (2025)
3. Akavia, A., Gentry, C., Halevi, S., Vald, M.: Achievable CCA2 relaxation for homomorphic encryption. In: TCC. pp. 70 – 99 (2022)
4. Alexandru, A., Badawi, A.A., Micciancio, D., Polyakov, Y.: Application-aware approximate homomorphic encryption: Configuring FHE for practical use. Tech. Rep. 203, IACR ePrint (2024)
5. Alexandru, A., Badawi, A.A., Micciancio, D., Polyakov, Y.: Application-aware approximate homomorphic encryption: Configuring FHE for practical use. IACR Communications in Cryptology **2**(4) (2025)
6. Alexandru, A., Kim, A., Polyakov, Y.: General functional bootstrapping using CKKS. In: CRYPTO. pp. 304–337 (2025)
7. 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)
8. Badawi, A.A., Alexandru, A., Bates, J., Bergamaschi, F., Cousins, D.B., Erabelli, S., Genise, N., Halevi, S., Hunt, H., Kim, A., Lee, Y., Liu, Z., Micciancio, D., Pascoe, C., Polyakov, Y., Quah, I., Saraswathy, R.V., Rohloff, K., Saylor, J., Suponitsky, D., Triplett, M., Vaikuntanathan, V., Zucca, V.: OpenFHE: Open-source fully homomorphic encryption library. Tech. Rep. ePrint 2022/915, IACR (2022)
9. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: CRYPTO. pp. 26–45 (1998)

10. Bendlin, R., Damgård, I.: Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In: TCC. pp. 201–218 (2010)
11. Bergamaschi, F., Costache, A., Dachman-Soled, D., Kippen, H., LaBuff, L., Tang, R.: Revisiting the security of approximate FHE with noise-flooding countermeasures. In: PKC. pp. 102–134 (2025)
12. Bergamaschi, F., Costache, A., Dachman-Soled, D., Kippen, H., LaBuff, L., Tang, R.: Revisiting the security of approximate FHE with noise-flooding countermeasures. In: PKC (2025)
13. Bernard, O., Joye, M., Smart, N.P., Walter, M.: Drifting towards better error probabilities in fully homomorphic encryption schemes. In: EUROCRYPT. pp. 181–211 (2025)
14. Biasse, J.F., Espitau, T., Fouque, P.A., Gélin, A., Kirchner, P.: Computing generator in cyclotomic integer rings: A subfield algorithm for the principal ideal problem in $l_{|\Delta_{\mathbb{K}}|}(1/2)$ and application to the cryptanalysis of a FHE scheme. In: EUROCRYPT. pp. 60–88 (2017)
15. Bogdanov, A., Lee, C.H.: Homomorphic encryption from codes. Tech. Rep. ePrint 2011/622, IACR (2011)
16. 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)
17. Boneh, D., Segev, G., Waters, B.: Targeted malleability: homomorphic encryption for restricted computations. In: ITCS. pp. 350–366 (2012)
18. Bonte, C., Iliashenko, I., Park, J., Pereira, H.V.L., Smart, N.P.: FINAL: faster FHE instantiated with NTRU and LWE. In: ASIACRYPT. pp. 188–215 (2022)
19. Bossuat, J.P., Costache, A., Mouchet, C., Nürnberger, L., Troncoso-Pastoriza, J.R.: Accurate and composable noise estimates for ckks with application to exact he computation. IACR Communications in Cryptology **2**(2) (2025)
20. Boudgoust, K., Scholl, P.: Simple threshold (fully homomorphic) encryption from lwe with polynomial modulus. In: ASIACRYPT (2023)
21. Brakerski, Z.: When homomorphism becomes a liability. In: TCC. p. 143–161 (2013)
22. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: CRYPTO. pp. 868–886 (2012)
23. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. TOCT pp. 1–36 (2014)
24. Branco, P., Malavolta, G., Maradni, Z.: Fully-homomorphic encryption from lattice isomorphism. In: TCC. pp. 220–252 (2025)
25. 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 Communications in Cryptology **2**(1) (2025)
26. Bultel, J.P., Checri, M., Fontaine, C., Renard, M., Sirdey, R., Stan, O.: Achieving CPA^D security for BFV: a pragmatic approach. Tech. Rep. 2288, IACR ePrint (2025)
27. Canetti, R., Raghuraman, S., Richelson, S., Vaikuntanathan, V.: Chosen-ciphertext secure fully homomorphic encryption. In: PKC. pp. 213–240 (2017)
28. Canonne, C.L., Kamath, G., Steinke, T.: The discrete gaussian for differential privacy. In: NeurIPS. p. xxx (2020)
29. Checri, M., Clet, P.E., Renard, M., Sirdey, R.: Achieving "beyond CCA1" security for linearly homomorphic encryption, without SNARKs? Tech. Rep. ePrint 2025/894, IACR (2025)

30. 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)
31. Cheon, J.H., Choe, H., Passelègue, A., Stehlé, D., Suvanto, E.: Attacks against the IND-CPAD security of exact FHE schemes. In: CCS. pp. 2505 – 2519 (2024)
32. Cheon, J.H., Hanrot, G., Kim, J., Stehlé, D.: Ship: A shallow and highly parallelizable CKKS bootstrapping algorithm. In: EUROCRYPT. pp. 398–428 (2025)
33. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: ASIACRYPT. pp. 409–437 (2017)
34. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: bootstrapping in less than 0.1 seconds. In: ASIACRYPT (2016)
35. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption over the torus. Journal of Cryptology (2019)
36. 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)
37. Cramer, R., Ducas, L., Peikert, C., Regev, O.: Recovering short generators of principal ideals in cyclotomic rings. In: EUROCRYPT. pp. 559–585 (2016)
38. Debris-Alazard, T., Fallahpour, P., Stehlé, D.: Quantum oblivious LWE sampling and insecurity of standard model lattice-based SNARKs. In: ACM STOC. pp. 423–434 (2024)
39. Debris-Alazard, T., Fallahpour, P., Stehlé, D.: Quantum oblivious LWE sampling and insecurity of standard model lattice-based SNARKs. Tech. Rep. ePrint 2024/030, IACR (2024)
40. Dwork, C., Naor, M., Reingold, O.: Immunizing encryption schemes from decryption errors. In: EUROCRYPT. pp. 342–360 (2004)
41. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Tech. Rep. 2012/144, IACR ePrint (2012)
42. Fontaine, C., Renard, M., Sirdey, R., Stan, O.: On new variants of funccpa security and related cca-secure constructions. Tech. Rep. ePrint 2025/2036, IACR (2025)
43. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: CRYPTO. pp. 75–92 (2013)
44. 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)
45. Leporati, A., Rovida, L., van Woerden, W.: Beyond LWE: a lattice framework for homomorphic encryption. Tech. Rep. ePrint 2025/1171, IACR (2011)
46. Li, B., Micciancio, D.: On the security of homomorphic encryption on approximate numbers. In: EUROCRYPT. pp. 648–677 (2021)
47. Li, B., Micciancio, D., Schultz, M., Sorrell, J.: Securing approximate homomorphic encryption using differential privacy. In: CRYPTO. pp. 560–589 (2022)
48. Li, B., Micciancio, D., Schultz-Wu, M., Sorrell, J.: Securing approximate homomorphic encryption using differential privacy. In: CRYPTO. p. 560–589 (2022)
49. Loftus, J., May, A., Smart, N.P., Vercauteren, F.: On CCA-secure somewhat homomorphic encryption. In: SAC. pp. 55–72 (2011)
50. Lyubashevsky, V., Palacio, A., Segev, G.: Public-key cryptographic primitives provably as secure as subset sum. In: TCC. p. 382–400 (2010)
51. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: EUROCRYPT. p. 1–23 (2010)
52. Manulis, M., Nguyen, J.: Fully homomorphic encryption beyond IND-CCA1 security: Integrity through verifiability. In: EUROCRYPT. pp. 63–93 (2024)

53. Micciancio, D., Suhl, A.: Simulation-secure threshold pke from lwe with polynomial modulus. Tech. Rep. 2023/1728, IACR ePrint (2023)
54. 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)
55. Nguyen, J.: GG-GSW: chosen-ciphertext secure leveled fhe from gadget trapdoors. Tech. Rep. ePrint 2026/316, IACR (2026)
56. Nguyen, J.: Personal communication (2026)
57. Ogilvie, T.: IND-CPA-D and KR-D security with reduced noise from the HintLWE problem. In: ASIACRYPT. p. 263–295 (2025)
58. Passelègue, A., Stehlé, D.: Low communication threshold fully homomorphic encryption. In: ASIACRYPT. pp. 297–329 (2024)
59. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC. pp. 84–93 (2005)
60. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM **56** (2009)
61. Schäge, S.: New limits of provable security and applications to ElGamal encryption. In: EUROCRYPT. p. 255–285 (2024)
62. Schäge, S., Vorstermans, M.: New limits for homomorphic encryption. In: ASIACRYPT. p. 482–515 (2025)
63. Walter, M.: What have SNARGs ever done for FHE? Tech. Rep. ePrint 2024/1207, IACR (2024)
64. Watanabe, Y., Shikata, J., Imai, H.: Equivalence between semantic security and indistinguishability against chosen ciphertext attacks. In: PKC. pp. 71–84 (2003)
65. Yang, R., Yu, Z., Susilo, W.: Fully homomorphic encryption with chosen-ciphertext security from LWE. In: CRYPTO. p. 371–405 (2025)

A Appendix sections organization

The appendix sections are organized as follows. First, Sect. B provides a few additional proofs that we have not included in the main body, including a proof (in Sect. B.3) that CPA^D security can be naturally achieved when given oracle access to the noise variance in (R)LWE ciphertexts. This latter result may be seen as a theoretical justification of the DEE heuristic that we attack in Sect. 5. Also, Sect. C presents more details on our KR^D attacks against DE-RLWE and DE-CKKS. Then Sect. D presents several folklore CCA1 attacks against Regev to which we refer to several times in the main body. Sect. E presents more details on some of the security notions considered in the main body or their relationships. In particular, this latter section provides the formal definition of (IND-)SA security [63] and its relationships with both CPA^D and vCCA^D security. Finally, Sect. F provides toy numerical experiments to help the reader picture the mechanics at work in our various attacks.

Lastly, we have also included a table of contents at the end of this document for convenience.

B Additional lemmas and proofs

B.1 Additional lemmas

Lemma 20.

$$\prod_{k=0}^{N-1}\left(1-\frac{1}{p^{N-k}}\right) \geq \left(1-\frac{1}{p}\right)^2.$$

Proof.

$$\begin{aligned}
\ln\left(\prod_{k=0}^{N-1}\left(1-\frac{1}{q^{N-k}}\right)\right) &= \sum_{k=1}^{N}\ln\left(1-\frac{1}{q^k}\right)\\
&= -\sum_{k=1}^{N}\sum_{j=1}^{+\infty}\frac{1}{jq^{jk}}\\
&= -\sum_{j=1}^{+\infty}\frac{1-\frac{1}{q^{jN}}}{j(q^j-1)}\\
&\geq -\sum_{j=1}^{+\infty}\frac{1}{j(q^j-1)}\\
&\geq -\sum_{j=1}^{+\infty}\frac{1}{j(\frac{q^j}{2})}\\
&\geq 2\ln\left(1-\frac{1}{q}\right)\\
&\geq \ln\left(\left(1-\frac{1}{q}\right)^2\right).
\end{aligned}$$

□

B.2 Alternative (direct) proof of Prop. 12

Proposition 21 (Same as Prop. 12). *$\mathcal{F}(\lambda)$-correct HELLHO schemes do not exist.*

Proof. We proceed by contradiction. Let,

$$\mathcal{E}_H = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, [\mathsf{AddPlain}, \mathsf{Mul}, \mathsf{MulByPlain}, ...]),$$

be a HELLHO scheme. Assume that $\mathcal{E}_H$ is correct over $\mathcal{F}(\lambda)$, i.e. that, $\forall m \in \mathcal{P}$ (following Eq. 1),

$$\mathcal{E}_H.\mathsf{Dec}(\mathcal{E}_H.\mathsf{Enc}(m; r)) = m, \tag{17}$$

and that, $\forall f \in \mathcal{F}(\lambda)$ and $\forall m_0, ..., m_{L-1} \in \mathcal{P}^L$, following Eq. (3),

$$\mathcal{E}_H.\mathsf{Dec}(\mathcal{E}_H.\mathsf{Eval}(f, \mathcal{E}_H.\mathsf{Enc}(m_0; r_0), ..., \mathcal{E}_H.\mathsf{Enc}(m_{L-1}; r_{L-1})) = f(m_0, ..., m_{L-1}), \tag{18}$$

both with overwhelming probability over the encryption randomness.

Assume $\mathcal{P} \supseteq \{0, 1\}$ and let

$$\mathsf{ct}_1 = \mathcal{E}_H.\mathsf{Enc}(1). \tag{19}$$

Now, consider a large enough set of encryptions of 0, $e_0, ..., e_{N-1}$, to form a generating set of $\mathbb{M}(\mathcal{C})$ (from Property 1 and Property 2 in Def. 4) and $\alpha \in \mathbb{Z}^N$ (from Property 3 in Def. 4) such that

$$\begin{aligned}\mathsf{ct}_1 &= \sum_{i=0}^{N-1} \alpha_i \cdot e_i, \\ &= \mathcal{E}_H.\mathsf{Add}(\mathsf{ScalMul}(\alpha_0, e_0), ..., \mathsf{ScalMul}(\alpha_{N-1}, e_{N-1})).\end{aligned}$$

Let $\oplus$ more conveniently denote $\mathcal{E}_H.\mathsf{Add}$. From Lemmas 8 and 9 (and, in particular, Eq. 7), we also have that

$$\mathsf{ct}_1 = \bigoplus_{i=0}^{N-1} \bigoplus_{j:\alpha_{ij}=1} e_{ij}, \tag{20}$$

where the α_{ij}'s are the digits in the base 2 decomposition of α_i, $e_{i,0} = e_i$ and $e_{ij} = e_{i,j-1} \oplus e_{i,j-1} \equiv \mathcal{E}_H.\mathsf{Add}(e_{i,j-1}, e_{i,j-1})$ $(j > 0)$. Remark that, still from Lemmas 8 and 9, the number of calls to $\mathcal{E}_H.\mathsf{Add}$ in Eq. (20) is polynomial in the security parameter so that (20) specifies a circuit in $\mathcal{F}(\lambda)$.

Then, from Eqs. (19) and (17), we have (with overwhelming probability) that

$$\mathcal{E}_H.\mathsf{Dec}(\mathsf{ct}_1) = 1.$$

However, because the e_i's are fresh encryptions of 0, Eqs. (20) and (17) yield that, with overwhelming probability,

$$\mathcal{E}_H.\mathsf{Dec}(\mathsf{ct}_1) = 0.$$

This thus contradicts the initial assumption that $\mathcal{E}_H$ is $\mathcal{F}(\lambda)$-correct, i.e. satisfies (17) and (18). □

B.3 CPA^D security with a "variance oracle"

Let us consider an FHE scheme based on Regev encryption/decryption for simplicity sake and assume that we are given a variance oracle $\mathsf{Var}^{\mathcal{O}} : \mathbb{Z}_q^{n+1} \longrightarrow \mathbb{R}_+$ such that, given a well-formed ciphertext ct under $\mathcal{R}$,

$$\mathsf{Var}^{\mathcal{O}}(\mathsf{ct}) \geq \sigma^2,$$

where σ^2 is the true variance of the noise in ct. Now consider the scheme $\mathcal{R}'$ obtained from $\mathcal{R}$ by modifying the decryption function as follows:

- $\mathcal{R}'.\mathsf{Dec}$, given $\mathsf{ct} = (a, b) \in \mathbb{Z}_q^{n+1}$ and $\sigma \in \mathbb{R}_+$, pick v uniformly in $[\![-B, B]\!]^k$ with $B = 2^\lambda \sigma \sqrt{2(\lambda+1)\log 2}$ (recall Lemma 3) and either:
  - (Option 1) Return $\perp$ when $|v| \geq \frac{\Delta}{2}$ and $\mathcal{R}.\mathsf{Dec}(\mathsf{ct})$ otherwise.
  - (Option 2) Return $\mathcal{R}.\mathsf{Dec}((a, b+v))$.

This definition is consistent in spirit to the definition of DE-RLWE in Sect. 5.1. The intuition behind these two options is that option 1 either returns $\perp$ or decrypts correctly with overwhelming probability. Conversely, in option 2, decryption may be incorrect because of the added smudging noise, but the decryption errors do not leak any information on the ciphertext noise. These two claims are established in the proof of Prop. 22 below.

Also remark that the CPA security of $\mathcal{R}'$ follows trivially from that of $\mathcal{R}$.

We now establish the easy result that given a variance oracle, $\mathcal{R}'$ is CPA^D secure which essentially provides the theoretical justification for the DEE heuristic in Sect. 5.1.

Proposition 22. *Given a CPA^D adversary against $\mathcal{R}'$ and a variance oracle $\mathsf{Var}^{\mathcal{O}} : \mathbb{Z}_q^{n+1} \longrightarrow \mathbb{R}_+$ such that, for a well-formed ciphertext ct under $\mathcal{R}'$, $\mathsf{Var}^{\mathcal{O}}(\mathsf{ct}) \geq \sigma^2$ where σ^2 is the true variance of the noise in ct, there exists an adversary $\mathcal{B}$ against the CPA security of $\mathcal{R}'$ that uses $\mathcal{A}$ as a subroutine.*

Proof. The reduction $\mathcal{B}$ maintains an internal state S of message-message-ciphertext triplets and then works as follows:

- An encryption request over $m \in \mathbb{Z}_t$ is transferred as is to the CPA challenger ($\mathcal{B}$ can even process it by itself in the public-key setting). Thus $\mathcal{B}$ obtains $\mathsf{ct} = \mathcal{R}'.\mathsf{Enc}(m)$, which it sends back to $\mathcal{A}$, after appending the triplet (m, m, ct) to S.
- A challenge request over $m_0, m_1 \in \mathbb{Z}_t^2$ is also transferred as is to the CPA challenger. Thus $\mathcal{B}$ obtains $\mathsf{ct} = \mathcal{R}'.\mathsf{Enc}(m_\gamma)$, for unknown bit γ, which it sends back to $\mathcal{A}$, after appending the triplet (m_0, m_1, ct) to S.
- When $\mathcal{A}$ issues an evaluation request over function $f \in \mathcal{F}(\lambda)$ and state indices $l_0, ..., l_{L-1}$, $\mathcal{B}$ handles it on its own by doing,

$$m'_0 = f(S[l_0].m_0, \ldots, S[l_{L-1}].m_0),$$

and

$$m'_1 = f(S[l_0].m_1, \ldots, S[l_{L-1}].m_1),$$

as well as

$$\mathsf{ct}' = \mathsf{Eval}(f, S[l_0].c, \ldots, S[l_{L-1}].c),$$

then giving ct' to $\mathcal{A}$ after appending the triplet $(m'_0, m'_1, \mathsf{ct}')$ to S.

– Finally, when $\mathcal{A}$ issues a decryption request over state index l, $\mathcal{B}$ first checks that $S[l].m_0 = S[l].m_1$ and returns $\perp$ when this is not the case. Otherwise, it invokes the variance oracle to get $\sigma^2 = \mathsf{Var}^{\mathcal{O}}(S[l].c)$, then picks v uniformly in $[\![-B, B]\!]^k$ with $B = 2^{\lambda}\sigma\sqrt{2(\lambda+1)\log 2}$ and returns either,
  - (Option 1) $\perp$ when $|v| \geq \frac{\Delta}{2}$, or $S[l].m_0$ (equivalently, $S[l].m_1$) otherwise.
  - (Option 2) $\lceil (\Delta S[l].m_0 + v)/\Delta \rfloor$ (equivalently, $\lceil (\Delta S[l].m_1 + v)/\Delta \rfloor$).

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,

– (Option 1) The reduction returns $\perp$ under the same condition that $\mathcal{R}'$ decryption function does. When this is not the case, since σ^2 is an upper bound for the true noise variance of e by definition of $\mathsf{Var}^{\mathcal{O}}$, the smudging lemma (Lemma 3) applies and $\mathcal{B}$ picks a smudging noise v such that $e + v$ is indistinguishable from v alone, i.e. such that $|e| << |v| < \frac{\Delta}{2}$ (with overwhelming probability). Hence, also with overwhelming probability, $\mathcal{R}.\mathsf{Dec}(S[l].c) = S[l].m_0 = S[l].m_1$. The claim for option 1 follows.
– (Option 2) For $S[l].c = (a, b)$, we have $b + v = \langle a, \mathsf{sk}\rangle + \Delta S[l].m_0 + e + v$ (recall $S[l].m_0 = S[l].m_1$ at this point in the reduction). Then,

$$\left\lceil \frac{1}{\Delta}((b+v) - \langle a, \mathsf{sk}\rangle) \right\rfloor = \left\lceil \frac{1}{\Delta}(\Delta S[l].m_0 + e + v) \right\rfloor,$$

is indistinguishable from $\left\lceil \frac{1}{\Delta}(\Delta S[l].m_0 + v) \right\rfloor$ by Lemma 3 (since σ^2 is an upper bound for the true noise variance of e by definition of $\mathsf{Var}^{\mathcal{O}}$). The claim thus follows also for Option 2.

□

As a last remark, we highlight that Option 1' below in $\mathcal{R}'.\mathsf{Dec}$ is equivalent to Option 1 above:

– (Option 1) Return $\perp$ when $|v| \geq \frac{\Delta}{2}$ and $\mathcal{R}.\mathsf{Dec}((a, b))$ otherwise.
– (Option 1') Return $\perp$ when $|v| \geq \frac{\Delta}{2}$ and $\mathcal{R}.\mathsf{Dec}((a, b + v))$ otherwise.

Indeed, given $\mathsf{ct} = (a, b) \in \mathbb{Z}_q^{n+1}$, with $b = \langle a, \mathsf{sk}\rangle + \Delta m + e$, and $|v| < \frac{\Delta}{2}$ we have:

– (Option 1) $|e| << |v| < \frac{\Delta}{2}$ (with overwhelming prob.), therefore (also with overwhelming prob.),

$$\left\lceil \frac{1}{\Delta}(b - \langle a, \mathsf{sk}\rangle) \right\rfloor = \left\lceil \frac{1}{\Delta}(\Delta m + e) \right\rfloor = m,$$

– (Option 1') $e + v \overset{i}{=} v$ (with overwhelming prob.), hence (also with overwhelming prob.),

$$\left\lceil \frac{1}{\Delta}(b + v - \langle a, \mathsf{sk}\rangle) \right\rfloor = \left\lceil \frac{1}{\Delta}(\Delta m + e + v) \right\rfloor \overset{i}{=} \left\lceil \frac{1}{\Delta}(\Delta m + v) \right\rfloor = m,$$

CPA^D security via ciphertext-independent noise variance monitoring. Consistently with the above, a popular approach to achieve CPA^D security from an (R)LWE-based scheme is via the technique of noise variance monitoring. To the best of our knowledge, it was introduced in [47] in the context of CKKS and easily adapts to other (R)LWE-based schemes. In a nutshell, the technique consists of starting from a baseline somewhat homomorphic (R)LWE scheme (which is most likely a HELLHO scheme following the discussion in Sect. 3.2) to define an extended scheme for which each ciphertext includes a bound for its noise variance. The scheme homomorphic operators are then also extended to update these bounds (recall that a CPA^D adversary by definition follows the cryptosystem specifications).

For example sake, we illustrate how this approach may be instantiated over the Regev scheme (Sect. 2.3) yielding the (linearly homomorphic) scheme Mon-$\mathcal{R}$ with plaintext domain $\mathbb{Z}_t$ and ciphertext domain $\mathbb{Z}_q^{n+1} \times \mathbb{R}_+$:

- Mon-$\mathcal{R}$.Enc: given $m \in \mathbb{Z}_t$, return $(c; \mathsf{v}) = (\mathcal{R}.\mathsf{Enc}(m); \sigma_0^2)$.
- Mon-$\mathcal{R}$.Add: given $(c_0; \mathsf{v}_0)$ and $(c_1; \mathsf{v}_1)$, both in $\mathbb{Z}_q^{n+1} \times \mathbb{R}_+$, return either:
  - (Average-case variance monitoring) $(\mathcal{R}.\mathsf{Add}(c_0, c_1); \mathsf{v}_0 + \mathsf{v}_1)$.
  - (Worst-case variance monitoring) $(\mathcal{R}.\mathsf{Add}(c_0, c_1); 4\max(\mathsf{v}_0, \mathsf{v}_1))$[25].
- Mon-$\mathcal{R}$.MulByPlain: given $(c; \mathsf{v}) \in \mathbb{Z}_q^{n+1} \times \mathbb{R}_+$ and $\alpha \in \mathbb{Z}_t$ return

$$(\mathcal{R}.\mathsf{MulByPlain}(\alpha, c); \alpha^2 \mathsf{v}).$$

- Mon-$\mathcal{R}$.Dec: given $(c; \mathsf{v})$ in $\mathbb{Z}_q^{n+1} \times \mathbb{R}_+$, return $\mathcal{R}'.\mathsf{Dec}(c, \sqrt{\mathsf{v}})$ (i.e. with smudging noise generation during decryption).

Because Mon-$\mathcal{R}$'s ciphertext domain is now $\mathbb{Z}_q^{n+1} \times \mathbb{R}_+$, Mon-$\mathcal{R}$ is *not* a HELLHO scheme, even if $\mathcal{R}$ is one (note also the non-linearity of Mon-$\mathcal{R}$.MulByPlain). Also, remark that when *worst-case* variance monitoring is used, the variance field of Mon-$\mathcal{R}$'s ciphertext is consistent with $\mathsf{Var}^{\mathcal{O}}$ above and, thus, Mon-$\mathcal{R}$ achieves CPA^D security. However, when *average-case* variance monitoring is used, smudging is not appropriately applied at decryption (since the variance field may significantly underestimate the true noise variance when ciphertexts with dependent noises have been homomorphically added). In this latter case, CPA^D security is thus jeopardized.

This also further illustrates the fact that, following the results in this paper, belonging to the class of HELLHO schemes is only a sufficient condition for CPA^D insecurity.

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

C Additional details on our KR^D attacks against DE-RLWE and DE-CKKS

C.1 Improved complexity analysis for the attacks in Sect. 5.2 and 5.3

DE-RLWE (Sect. 5.2). For simplicity sake, we only presented a crude complexity analysis (based on the generic reduction in the proof of Prop. 13) for the complexity of the KR^D attack against DE-RLWE of Sect. 5.2 (footnote 22, p. 26). However, it turns out that its complexity can be lowered from $O(\lambda^6)$ to $O(\lambda^4)$ by means of the following more specific approach.

Rather than solving one linear system per decryption request[26], the adversary can solve a *single* linear system over $\mathbb{Z}_q^{2n}$ (in $O(n^3 \log_2 q)$, assuming q is a prime power) to generate ciphertext $(0, ..., 0; 1, 0, ..., 0)$ following Lemma 9. From this ciphertext, it can build any ciphertext of the form,

$$(0, ..., 0; \alpha, 0, ..., 0), \tag{21}$$

with $\alpha \in [\![0, \Delta]\!]$ by means of logarithmically many calls to DE-RLWE.Add, following Lemma 8 (p. 14). As such, it can further do so by means of evaluation requests towards its KR^D challenger so that the above ciphertext ends up in its internal state (along with 0 left and right cleartext values).

Then, following the proof of Prop. 19, given $c_0 = (a, b)$, to perform a (KR^D) decryption request over ciphertext,

$$(a, b_0 + \alpha, b_1, ..., b_{k-1}, b_k, ..., b_{n-1}),$$

the adversary can simply sum c_0 with the ciphertext in Eq. (21) by means of a call to DE-RLWE.Add. Again, by doing so via an evaluation request, the above ciphertext ends up in the KR^D challenger's internal state (also along with 0 left and right cleartext values).

In doing so, the complexity of the attack is dominated by the resolution of the initial linear system and that of the final one (for recovering sk). Thus the attack complexity is in $O(n^3 \log_2 q)$ i.e. $O(\lambda^4)$ when n and q respectively are in $O(\lambda)$ and $O(2^\lambda)$.

DE-CKKS (Sect. 5.2). In a similar spirit, we can also do better than relying on the generic reduction in the proof of Prop. 13 for this attack (as already hinted in Sect. 5.3 on p. 27). Indeed, a KR^D adversary against DE-CKKS may proceed by first solving a *single* linear system over $\mathbb{Z}_q^{2n}$ (in $O(n^3 \log_2 q)$, assuming q is a prime power) to generate ciphertext,

$$(0, ..., 0; \underbrace{0, ..., 0}_{k}, \underbrace{1, 0, ..., 0}_{n-k}), \tag{22}$$

[26] Recall from Prop. 13 that each linear system resolution is done as a prerequisite to populate the KR^D challenger's internal state with a ciphertext the adversary wants to decrypt.

following Lemma 9. From this ciphertext, it can then build ciphertext,

$$(0, ..., 0; \underbrace{0, ..., 0}_{k}, \underbrace{[-B_0]_q, 0, ..., 0}_{n-k}), \tag{23}$$

by means of Lemma 8. For both ciphertexts, it does so by means of evaluation requests so that they both end up in its KR^D challenger's internal state (along with 0 left and right cleartext values).

Then, following the description of the attack in Sect. 5.3, given $c_0 = (a, b)$, to perform successive (KR^D) decryption requests over ciphertexts,

$$(a, b_0, ..., b_{k-1}, b_k + \alpha, ..., b_{n-1}), \tag{24}$$

for $\alpha \in [\![-B_0, B_0]\!]$, the adversary can simply sum c_0 with the ciphertext in Eq. (23) by means of a call to DE-RLWE.Add and then iteratively add ciphertext (22) (also by means of DE-RLWE.Add) to proceed with the attack loop. Again, by doing so via evaluation requests, the resulting ciphertexts end up in the KR^D challenger's internal state (also along with 0 as left and right cleartext values) which thus accepts decryption requests over them (or, rather, over the appropriate indices in the challenger's internal state).

It follows that the KR^D variant of the attack needs to perform only two linear system resolutions: the initial one for building ciphertext (23) and the final one for recovering sk. However, because the attack has a heuristic part, it is difficult to determine its worst-case complexity (although its concrete practicality is backed up by our experiments in Sect. 5.3 and Table 1).

C.2 Polynomial fitting procedure in the attack of Sect. 5.3

In the attack of Sect. 5.3 against DE-CKKS (please recall the notations in that section), the adversary looks for α^* such that u_α is minimum by iterating over $\alpha \in [\![-B_0, B_0]\!]$ and heuristically uses $\hat{u}_\alpha$ in lieu of u_α. On experimental grounds, using

$$\hat{\alpha}^* = \mathrm{argmin}_{\alpha \in [\![-B_0, B_0]\!]} \hat{u}_\alpha, \tag{25}$$

yields a robust enough estimate for the noise in the attacked noise slot only at the cost of many supplementary decryptions[27] (an order of magnitude of 100 to 150 for the parameters in Table 1). For example for $n = 8192$, $k = 4096$ and 150 decryptions per ciphertext, with a parallel implementation running on the 20 HW threads of a highend laptop PC (the same that we used for the last column of Table 1), we observed 5 erroneous noise predictions over 10000 trials leading to an overall attack success probability of around $(1 - 5/10000)^{8192} \approx 2^{-6}$ for around 13 hours of wallclock time. While this is enough to yield meaningful attacks, we managed to highly improve the reliability of the noise prediction by

[27] Recall that the adversary may arbitrarily increase the size of the sample it can observe by asking several decryptions of the same ciphertext, since the decryption function picks a fresh smudging noise each time it is called.

means of an elementary degree-two polynomial regression that we present below for completeness[28]. As shown in the last column of Table 1, this optimization allows an adversary to perform the attack for the above parameters in only 6 mins in the same conditions.

So the rest of this section covers very elementary aspects that we included in order for our experimental results to be reproducible (we also provided the proof of concept code for the attack alongside the submission).

To make the estimation more robust we thus heuristically model the relationship between α and u_α as a degree two polynomial, i.e. heuristically assume that,

$$u_\alpha = a\alpha^2 + b\alpha + c.$$

Then, given $\hat{u}_\alpha$ for $\alpha \in [\![-B_0, B_0]\!]$, we look for a, b and c such that,

$$\varepsilon(a,b,c) = \frac{1}{2}\sum_{\alpha=-B_0}^{B_0}(a\alpha^2 + b\alpha + c - \hat{u}_\alpha)^2,$$

is minimum. We then proceed the usual way and compute,

$$\begin{aligned}
\frac{\partial\varepsilon}{\partial a} &= a\sum_{\alpha=-B_0}^{B_0}\alpha^4 + b\underbrace{\sum_{\alpha=-B_0}^{B_0}\alpha^3}_{=0} + c\sum_{\alpha=-B_0}^{B_0}\alpha^2 - \sum_{\alpha=-B_0}^{B_0}\hat{u}_\alpha\alpha^2,\\
\frac{\partial\varepsilon}{\partial b} &= a\underbrace{\sum_{\alpha=-B_0}^{B_0}\alpha^3}_{=0} + b\sum_{\alpha=-B_0}^{B_0}\alpha^2 + c\underbrace{\sum_{\alpha=-B_0}^{B_0}\alpha}_{=0} - \sum_{\alpha=-B_0}^{B_0}\hat{u}_\alpha\alpha,\\
\frac{\partial\varepsilon}{\partial c} &= a\sum_{\alpha=-B_0}^{B_0}\alpha^2 + b\underbrace{\sum_{\alpha=-B_0}^{B_0}\alpha}_{=0} + c(2B_0+1) - \sum_{\alpha=-B_0}^{B_0}\hat{u}_\alpha.
\end{aligned}$$

Equating the partial derivatives to 0 thus yields a linear system that we solve to get the values, $\hat{a}$, $\hat{b}$ and $\hat{c}$ that minimize ε. Finally, we use

$$-\frac{\hat{b}}{2\hat{a}},$$

as an estimate for α^*, and thus $\left\lfloor \frac{\hat{b}}{2\hat{a}} \right\rceil$ as a guess for the target noise. This yielded the experimental results in Table 1.

[28] However, we also observed that Eq. (25) provides a more robust estimate than our polynomial regression when the number of noise slots is very small (typically less than 100). Still, this parameter regime is *not* relevant since the estimation of the ciphertext noise variance which is used to drive smudging is then done on a very small sample and, hence, is not reliable.

C.3 Porting the attack of Sect. 5.3 to DE-CKKS with encoding (sketch)

In this section, we sketch a plausible strategy to adapt the attack of Sect. 5.3 against DE-CKKS (with direct encoding) to the case where CKKS' FFT-based encoding is used.

Recall the DFT, $\tilde{x} = \mathsf{DFT}(x)$, for $x \in \mathbb{C}^n$, $\tilde{x}_\ell = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} x_k e^{-\frac{2i\pi k\ell}{n}}$, and its inverse, $x_k = \frac{1}{\sqrt{n}} \sum_{\ell=0}^{n-1} \tilde{x}_\ell e^{\frac{2i\pi k\ell}{n}}$.

In CKKS (assuming private key to simplify), encryption does, given $m \in \mathbb{C}^{n/2}$,

$$(a, a \cdot \mathsf{sk} + e + \lfloor \mathsf{iDFT}(\Pi^{-1}(m)) \rceil), \tag{26}$$

with $\Pi^{-1} : \mathbb{C}^{n/2} \longrightarrow \mathbb{C}^n$, such that,

$$\Pi^{-1}(m) = (m_0, ..., m_{n/2-1}, m_0^*, ..., m_{n/2-1}^*),$$

where z^* denotes the conjugate of z. In particular, this implies that $\mathsf{iDFT}(\Pi^{-1}(m)) \in \mathbb{R}^n$. Consistently, $\Pi : \mathbb{C}^n \longrightarrow \mathbb{C}^{n/2}$ just drops the second half of its input vector.

Since it is standard to model the rounding error as uniformly random in $[-1/2, 1/2)$ [ePrint-2024/853], there will be more important sources of imprecision in the following attack sketch (due to the use of statistical estimations). Hence, we abstract out the rounding in Eq. (26) for the remainder of this section. We also abstract out the scaling factor Δ in this sketch, to lighten the notations.

Contrary to standard CKKS encryption, DE-CKKS's encryption takes $m \in \mathbb{R}^{n/2}$ (rather than $\mathbb{C}^{n/2}$) and then proceeds normally following Eq. (26). Regarding decryption, DE-CKKS.Dec first does,

$$m = \Pi(\mathsf{DFT}(b - a \cdot \mathsf{sk})) \in \mathbb{C}^{n/2},$$

and then use $\hat{\sigma}^2 = \frac{2}{n} \|\Im(m)\|_2^2$ to return $\Re(m) + \upsilon(\hat{\sigma}^2)$, where $\upsilon(\hat{\sigma}^2)$ is a smudging noise vector (that may be iid gaussian or uniform). Generally the smudging noise variance is $\propto 2^\lambda \hat{\sigma}^2$.

Now, consider an encryption of 0, (a, b) with $b = a \cdot \mathsf{sk} + e$, let $\mu = b - a \cdot \mathsf{sk}$ and $\iota = \Im(\Pi(\mathsf{DFT}(\mu))) = \Im(\Pi(\mathsf{DFT}(e)))$, and suppose that we want to add a value α to ι_j. This can be done by computing $b'_{\alpha,j} = b + \mathsf{iDFT}(\Pi^{-1}(\alpha i u_j))$, where u_j is the j-th standard basis vector. Then the variance used as a basis for smudging during the decryption of ciphertext $(a, b'_{\alpha,j})$ is (similarly to Eq. 16 in Sect. 5.3),

$$\hat{\sigma}^2 = \frac{2}{n} \left((\iota_j + \alpha)^2 + \sum_{k=0, k\neq j}^{n/2-1} \iota_k^2 \right),$$

and is minimum for $\alpha = -\iota_j$. We may then use an optimization procedure like those we used in the attack of Sect. 5.3 to obtain an estimate $\hat{\iota}_j$ of ι_j (from the value α^* that minimizes the empirical variance of $\Re(\Pi(\mathsf{DFT}(\mu'_{\alpha,j})))$, $\mu'_{\alpha,j} = b'_{\alpha,j} - a \cdot \mathsf{sk}$, observed by doing statistics over several calls to the decryption function over $(a, b'_{\alpha,j})$). Let $\hat{\iota}$ be the estimate obtained by repeating this process over $j \in [\![0, n/2 - 1]\!]$, i.e. $\hat{\iota} \approx \Im(\Pi(\mathsf{DFT}(e)))$, and do, $b' = b - \mathsf{iDFT}(\Pi^{-1}(i\hat{\iota}))$.

Assume, for the sake of the argument, that $\hat{\iota} = \iota = \mathfrak{I}(\Pi(\mathsf{DFT}(e)))$ (i.e. that the estimation is correct), in this case we have $\mathfrak{I}(\mathsf{DE\text{-}CKKS.Dec}((a,b'))) = 0$. Indeed, let $\mu' = b' - a \cdot \mathsf{sk}$,

$$\begin{aligned}
\Pi(\mathsf{DFT}(\mu')) &= \Pi(\mathsf{DFT}(e - \mathsf{iDFT}(\Pi^{-1}(i\hat{\iota})))),\\
&= \Pi(\mathsf{DFT}(e)) - \Pi(\mathsf{DFT}(\mathsf{iDFT}(\Pi^{-1}(i\hat{\iota})))),\\
&= \Pi(\mathsf{DFT}(e)) - i\hat{\iota},\\
&= \Pi(\mathsf{DFT}(e)) - i\mathfrak{I}(\Pi(\mathsf{DFT}(e))),\\
&= \mathfrak{R}(\Pi(\mathsf{DFT}(e))).
\end{aligned} \tag{27}$$

Therefore, in this particularly favorable thought experiment, smudging is cancelled and $\mathsf{DE\text{-}CKKS.Dec}((a,b'))$ then allows one to recover $\mathfrak{R}(\Pi(\mathsf{DFT}(e)))$. Hence, still in the favorable setup, we get,

$$e = \mathsf{iDFT}(\Pi^{-1}(\mathsf{DE\text{-}CKKS.Dec}((a,b')) + i\hat{\iota})), \tag{28}$$

thus yielding the sought-after linear system allowing an adversary to recover sk.

Overall, Eqs. (27) and (28) suggest that, even when we only have that $\hat{\iota} \approx \mathfrak{I}(\Pi(\mathsf{DFT}(e)))$, the smudging variance is significantly reduced and,

$$\mathsf{DE\text{-}CKKS.Dec}((a,b')) \approx \mathfrak{R}(\Pi(\mathsf{DFT}(e))).$$

Then,

$$\hat{e} = \mathsf{iDFT}(\Pi^{-1}(\mathsf{DE\text{-}CKKS.Dec}((a,b')) + i\hat{\iota})) \approx e,$$

and we may thus hope that,

$$(a, b - \hat{e}),$$

yields an easier RLWE instance (i.e. with a much smaller noise than the original).

Needless to emphasize that this attack strategy is more intricate than the one we describe in the simpler case where the encoding is abstracted out and would (just like the attack in Sect. 5.3) deserve experimental validation to assess its practical efficiency. With that respect, we speculate that doing so would allow one to observe significant drops in the security margin (measured e.g. with the lattice-estimator) rather than concrete complete recovery of keys.

Lastly, if the above technique eventually allows one to obtain an estimation $\hat{e}$ such that $\|e - \hat{e}\| \leq \varepsilon$ for some $\varepsilon > 0$ and given an upper bound B on $\|e\|_\infty$ satisfied with overwhelming probability (such a bound is easy to obtain for fresh ciphertexts), we may further be able to iteratively improve the precision of the estimation by applying it again to the rescaled ciphertext,

$$B/\varepsilon \cdot (a, b - \hat{e}).$$

Let $e' = e - \hat{e}$ and $\hat{e}'$ denote the new estimation obtained for e'. Then,

$$\|(B/\varepsilon)(e - \hat{e}) - \hat{e}'\| \leq \varepsilon,$$

and, hence,

$$\|e - (\hat{e} + \hat{e}' \cdot (\varepsilon/B))\| \leq \frac{\varepsilon^2}{B},$$

yielding a more precise estimation. Thus, applying this rescaling step several times may plausibly help improving the practicality of the attack.

D Folklore CCA1 attacks against Regev

In this section, we briefly present two lines of folklore CCA1 attacks against the Regev scheme.

D.1 A non-adaptive "ill-formed" ciphertext-based attack

As discussed in Sect. 2.3, let us remark that the "ill-formed"ciphertext $(-\Delta\mathbf{1}_i, 0)$ decrypts to $[\mathsf{sk}_i]_t$, where $\mathbf{1}_i$ is the i-th standard basis vector. As a slight simplification, let us assume that $\Delta = \frac{q}{t} = \frac{2^k}{2^l} = 2^{k-l}$ and ask for the decryption of ill-formed ciphertext $c_l = \left(-\frac{\Delta}{t}\mathbf{1}_i, 0\right) = (-2^{k-2l}\mathbf{1}_i, 0)$ which gives $\left[\left\lceil \frac{\mathsf{sk}_i}{t} \right\rfloor\right]_t$. Since $\frac{\mathsf{sk}_i}{t} = \left\lfloor \frac{\mathsf{sk}_i}{t} \right\rfloor + \frac{[\mathsf{sk}_i]_t}{t}$, two cases can then occur,

- When $\frac{[\mathsf{sk}_i]_t}{t} < \frac{1}{2}$ (recall that we know $[\mathsf{sk}_i]_t$ from above), $\mathcal{R}.\mathsf{Dec}(c_l) = \left[\left\lceil \frac{\mathsf{sk}_i}{t} \right\rfloor\right]_t = \left[\left\lfloor \frac{\mathsf{sk}_i}{t} \right\rfloor\right]_t$ and the decryption of c_l straightaway gives us bits $l, l+1, ..., 2l-1$ of sk_i.
- When $\frac{[\mathsf{sk}_i]_t}{t} \geq \frac{1}{2}$, $\mathcal{R}.\mathsf{Dec}(c_l) = \left[\left\lceil \frac{\mathsf{sk}_i}{t} \right\rfloor\right]_t = \left[\left\lfloor \frac{\mathsf{sk}_i}{t} \right\rfloor + 1\right]_t$. So either $\mathcal{R}.\mathsf{Dec}(c_l) = 0$ and thus $\left[\left\lfloor \frac{\mathsf{sk}_i}{t} \right\rfloor\right]_t = t-1$, or $\left[\left\lfloor \frac{\mathsf{sk}_i}{t} \right\rfloor\right]_t = \mathcal{R}.\mathsf{Dec}(c_l)-1$ otherwise. So $\mathcal{R}.\mathsf{Dec}(c_l)-1 \mod t$ gives away bits $l, l+1, ..., 2l-1$ of sk_i.

A CCA1 adversary may then further proceeds similarly by exploiting ciphertext $c_{2l} = \left(-\frac{\Delta}{t^2}\mathbf{1}_i, 0\right) = (-2^{k-3l}\mathbf{1}_i, 0)$ and its knowledge of $[\mathsf{sk}_i]_{t^2}$ (from above) to retrieve bits $2l, 2l+1, ..., 3l-1$ of sk_i, and so on and so forth. Let us emphasize that this attack is non-adaptive.

D.2 An adaptive noise recovery attack

Another line of folklore attacks follows a dichotomic search pattern (which has been used numerous times, e.g. [49,30]) where the adversary starts from a well-formed encryption of 0, $c_0 = (a, b)$ with $b = \langle a, \mathsf{sk}\rangle + e$, to find the critical value α^* such that $\mathsf{Dec}(a, b+\alpha^*) \neq 0$, i.e. such that $e + \alpha^* = \frac{\Delta}{2}$. The adversary may then conclude that the noise in c_0 is equal to $\frac{\Delta}{2} - \alpha^*$, thereby getting one linear equation in sk. By repeating this (adaptive) process over n encryptions of 0, the adversary can eventually retrieve sk by means of linear algebra techniques.

Remark that this attack does not use the legitimate add-by-const operator which only allows one to add multiples of Δ to the b-term of an LWE pair.

E Additional details on security notions

E.1 CPA, CCA1 and CCA2 security.

Definition 23 (Generic indistinguishability experiment). *Given an encryption scheme* $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, [...])$ *and a PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$

with state S, we define the following generic indistinguishability experiment, $\mathsf{Exp}_{\mathcal{E}}^{\mathcal{O}_1,\mathcal{O}_2}(\mathcal{A})$,

$$
\begin{aligned}
&(\mathsf{ek},\mathsf{sk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda),\\
&(m_0,m_1,S) \leftarrow \mathcal{A}_1^{[\mathcal{E}.\mathsf{Enc}(\mathsf{ek},.)],\mathcal{O}_1}([\mathsf{ek}]),\\
&\gamma \xleftarrow{\$} \{0,1\},\\
&c^* \leftarrow \mathcal{E}.\mathsf{Enc}(m_\gamma),\\
&\gamma' \leftarrow \mathcal{A}_2^{[\mathcal{E}.\mathsf{Enc}(\mathsf{ek},.)],\mathcal{O}_2}([\mathsf{ek}],S,c^*),\\
&\gamma = \gamma',
\end{aligned}
$$

where $\mathcal{O}_1$ and $\mathcal{O}_2$ respectively denote first and second step decryption oracles and with an encryption oracle $\mathcal{E}.\mathsf{Enc}(\mathsf{ek},.)$ being provided only in the private key case.

Then, for a PPT adversary, $\mathcal{A} = (\mathcal{A}_1,\mathcal{A}_2)$, the CPA and CCA1 experiments, respectively $\mathsf{Exp}_{\mathcal{E}}^{\mathrm{CPA}}(\mathcal{A})$ and $\mathsf{Exp}_{\mathcal{E}}^{\mathrm{CCA1}}(\mathcal{A})$, respectively correspond to $\mathsf{Exp}_{\mathcal{E}}^{\cdot,\cdot}(\mathcal{A})$ and $\mathsf{Exp}_{\mathcal{E}}^{\mathcal{E}.\mathsf{Dec}(\mathsf{sk},.),\cdot}(\mathcal{A})$. The CCA2 experiment is also defined as,

$$\mathsf{Exp}_{\mathcal{E}}^{\mathrm{CCA2}}(\mathcal{A}) \equiv \mathsf{Exp}_{\mathcal{E}}^{\mathcal{E}.\mathsf{Dec}(\mathsf{sk},.),\mathcal{O}_{\mathrm{CCA2}}(\mathsf{sk},c^*,.)}(\mathcal{A}),$$

with

$$\mathcal{O}_{\mathrm{CCA2}}(c^*,c) = \begin{cases} \mathcal{E}.\mathsf{Dec}(c) & \text{if } c^* \neq c,\\ \perp & \text{otherwise.}\end{cases}$$

A scheme $\mathcal{E}$ is then said to be CPA secure if for any PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}_{\mathcal{E}}^{\mathrm{CPA}}(\mathcal{A}) = \left|P\left(\mathsf{Exp}_{\mathcal{E}}^{\mathrm{CPA}}(\mathcal{A}) = \mathsf{True}\right) - 1/2\right| \leq \mathrm{negl}(\lambda),$$

where λ is the security parameter, and likewise for CCA1 and CCA2.

In the general regime where approximate or somewhat correct FHE schemes are allowed, we also have the following separation [25]:

$$\mathrm{CPA} \prec \mathrm{CPA}_1^D \prec \mathrm{CCA1}. \tag{29}$$

E.2 Application-aware Homomorphic Encryption(AAHE)

As already said in Sect. 2.2, following several CPA^D attacks against either "exact" and approximate schemes [30,31,44], Alexandru et al. [4,5] have proposed a new *weaker* variant of CPA^D security, termed Application-Aware HE security, that we denote AAHE to avoid confusion. In essence, this new definition formalizes that a meaningful CPA^D security notion can also be defined relative to a circuit subset and a noise estimation strategy (which then serves, e.g., to implement the noise flooding mechanism in [47]). With respect to that new security notion, the cryptosystem parameters should then be set relative to these, and the homomorphic evaluations should (somehow) be limited to the functions or circuits in the class.

For example, if we consider again a CPA secure HE scheme over circuit class $\mathcal{F}(\lambda)$ that achieves correctness over $\mathcal{H} \subset \mathcal{F}(\lambda)$, then that scheme is $\mathcal{H}$-AAHE secure (by direct reduction to its CPA security) while it may not at all achieve CPA^D security. Beyond this simple example, an approximate or somewhat-correct scheme that is AAHE secure over circuit class $\mathcal{H}$ with respect to an average-case noise estimation strategy (which is valid when the input ciphertexts to Eval have independent noises) may not be secure when ciphertexts with dependent noises are instead used as inputs. Interestingly, to avoid manual enforcement of the above constraints, Alexandru et al. [4] further proposes an approach based on an Application Specification Language (ASL). Using the ASL, the users may then describe its application in a way that can easily be parsed by the library to determine the appropriate scheme parameters.

E.3 vCCA^D security

Following [52], [25] also defines the notion of vCCA^D security, which, in a nutshell, is a "CPA^D-style" multiple challenge variant of vCCA in which the decryption oracle also accepts byproducts of the challenge ciphertexts as long as the associated left and right cleartext evaluations coincide.

Still following [25], vCCA and vCCA^D security are related as follows. *Under the $\mathcal{F}(\lambda)$-correctness assumption*, the two notions are equivalent. In the general regime where approximate or somewhat correct FHE are allowed, we have that vCCA $\prec$ vCCA$^D_{\mathrm{SC}} \prec$ vCCA^D, where vCCA$^D_{\mathrm{SC}}$ is the single-challenge variant of vCCA^D. Additionally, still in that latter regime, vCCA security remains equivalent to its multiple-challenge variant.

In the private key setting[29], the vCCA^D game decryption oracle is defined as:

- Decryption request: given a ciphertext c the vCCA^D challenger proceeds as follows. Let $(f, c_0, ..., c_{L-1}) = \mathsf{extract}(c)$. If,

$$f(\mathrm{left}(c_0), ..., \mathrm{left}(c_{L-1})) \neq f(\mathrm{right}(c_0), ..., \mathrm{right}(c_{L-1})), \tag{30}$$

  then return $\perp$ to $\mathcal{A}$. Otherwise, return $\mathsf{Dec}(c)$.

Where the vCCA^D challenger maintains an internal state S (similar to that of a CPA^D challenger) and where for any ciphertext $c \in \mathcal{C}$ we define[30]

$$\mathrm{left}(c) = \begin{cases} S[i].m_0 \text{ if } \exists i : S[i].c = c, \\ \perp \qquad\quad \text{otherwise,} \end{cases} \tag{31}$$

as well as,

$$\mathrm{right}(c) = \begin{cases} S[i].m_1 \text{ if } \exists i : S[i].c = c, \\ \perp \qquad\quad \text{otherwise,} \end{cases} \tag{32}$$

[29] In the public key setting, the definition of vCCA^D security further has to assume the existence of a plaintext extractor. We refer the reader to [25, Sect. 3.3.1] regarding this minor technical subtlety.

[30] With the convention that $f(m_0, ..., m_{L-1}) = \perp$ when $\exists i : m_i = \perp$ so that if the left and right evaluations both give $\perp$, condition (30) is not satisfied and $\mathsf{Dec}(c)$ is returned to $\mathcal{A}$.

E.4 (IND-)SA security

The (IND-)SA security notion has been introduced in [63] to provide a meaningful security model for constructions providing integrity guarantees to homomorphic evaluations, for example constructions that associate an FHE scheme to a SNARG or SNARK for proving genuine homomorphic computations.

The (IND-)SA game [63] is strongly inspired by the CPA^D one. Rather than providing an evaluation oracle, as in the CPA^D game, the decryption oracle takes as inputs a ciphertext $\mathsf{ct} \in \mathcal{C}$, a function $f \in \mathcal{F}(\lambda)$ as well as a number of indices $i_0, ..., i_{l-1}$ pointing to the challenger internal state S (which is then filled by means only of the encryption oracle and thus contains only fresh well-formed ciphertexts). Furthermore, the scheme decryption function is assumed to be given an "extended" ciphertext containing both the ciphertext to be decrypted as well as a function and some input ciphertexts that may explain the ciphertext to decrypt (for this reason, the security notion does not rule out trivial non-compact FHE). The decryption oracle then proceeds as follows:

- If $\exists j \in [\![0, l-1]\!] : i_j \geq |S|$, then return $\perp$.
- If
$$f(S[i_0].m_0, ..., S[i_{l-1}].m_0) \neq f(S[i_0].m_1, ..., S[i_{l-1}].m_1), \tag{33}$$
then return $\perp$.
- Otherwise, return $\mathcal{E}_H.\mathsf{Dec}(\mathsf{ct}, f, S[i_0].c, ..., S[i_{l-1}].c)$.

In [63], (IND-)SA security is also introduced and studied under statistical $\mathcal{F}(\lambda)$-correctness but the notion equally makes sense when such an assumption does not hold. Because of this, the relationship between (IND-)SA and CPA^D was not investigated in [63]. In the next proposition, we thus establish the fact that (IND-)SA implies CPA^D by means of a simple reduction.

Proposition 24 ((IND-)SA$\Longrightarrow$CPA^D). *If there exists an adversary $\mathcal{A}$ against the CPA^D security of scheme $\mathcal{E}$, then there exists an adversary $\mathcal{B}$ against the (IND-)SA security of $\mathcal{E}$.*

Proof. The reduction forwards encryption and challenge requests as is to its (IND-)SA challenger. It forwards the resulting ciphertexts to $\mathcal{A}$, stores them in a state of its own and maintains an index conversion table between its challenger's view and that of $\mathcal{A}$. When $\mathcal{A}$ issues an evaluation request over function $f \in \mathcal{F}(\lambda)$ and state indices $l_0, ..., l_{L-1}$, $\mathcal{B}$ processes it on its own and returns the resulting ciphertext to $\mathcal{A}$ after storing it, along with function f and the state indices in the request. Then, when $\mathcal{A}$ issues a decryption request over state index l, $\mathcal{B}$ determines from its internal memory whether it targets a fresh or evaluated ciphertext. In case of a fresh ciphertext, ct, stored at the corresponding index (from $\mathcal{A}$'s view), $\mathcal{B}$ issues a decryption request with parameters $(\mathsf{ct}, \mathsf{id}, l')$, with l' denoting the index of ciphertext ct is the challenger's view, towards its (IND-)SA challenger and forwards the result to $\mathcal{A}$. In case of an evaluated ciphertext, ct, $\mathcal{B}$ retrieves function f and the ($\mathcal{A}$'s view) fresh ciphertexts indices (wlog we assume one-hop FHE evaluations) that served as inputs to the corresponding evaluation

request, $l_0, ..., l_{L-1}$. Denoting by $l'_0, ..., l'_{L-1}$ the corresponding challenger's view indices, it then issues a decryption request with parameters $(\mathsf{ct}, f, l'_0, ..., l'_{L-1})$ to its (IND-)SA challenger and finally forwards the result to $\mathcal{A}$.

The reduction then works since the (IND-)SA decryption oracle does the same verification as the CPA^D one w. r. t. equality of the left and right cleartext evaluations. □

We also provide the following proposition that establishes that vCCA^D implies (IND-)SA security in order to complete Fig. 1.

Proposition 25 (vCCA^D$\Longrightarrow$(IND-)SA). *If there exists an adversary $\mathcal{A}$ against the (IND-)SA security of scheme $\mathcal{E}$, then there exists an adversary $\mathcal{B}$ against the vCCA^D security of $\mathcal{E}$.*

Proof. Consider the reduction where $\mathcal{B}$ forwards encryption and decryption requests from $\mathcal{A}$ as they are to its vCCA^D challenger and maintains an internal state, $S^{\mathcal{B}}$, for storing the resulting ciphertexts (the reduction does not have to store the associated left and right cleartext values which end up stored in the internal state of the vCCA^D challenger). Then the reduction handles a decryption request over ciphertext $\mathsf{ct} \in \mathcal{C}$, function $f \in \mathcal{F}(\lambda)$ and indices $l_0, ..., l_{L-1}$ simply by transferring the "extended" ciphertext,

$$(\mathsf{ct}, f, S^{\mathcal{B}}[l_0], ..., S^{\mathcal{B}}[l_{L-1}]),$$

as a decryption request to its vCCA^D challenger (recall from above that the scheme's decryption function expects such an "extended" ciphertext).

This reduction works[31] since the vCCA^D decryption oracle retrieves both the function f and the input ciphertexts (by means of its extractor) as well as the input messages (by means of its internal state) and performs the same verification over the left and right cleartext values (Eq. 30) that is expected from an (IND-)SA challenger (Eq. 33). □

Note that [63] establishes that CPA<(IND-)SA<vCCA but *under $\mathcal{F}(\lambda)$-correctness*. However, (IND-)SA is a multiple challenge notion. Hence, in the general regime where this assumption does not hold, by analogy with the results in [25], it is likely that vCCA does not imply (IND-)SA but only a strictly weaker single-challenge variant of it. We leave these minor additional questions open (although we think that they are not hard to settle using the techniques in [25]).

E.5 Ciphertext pseudorandomness vs IND-CPA

In this section, we separate the notions of ciphertext pseudorandomness (well-formed ciphertexts under a given scheme are indistinguishable from uniform over its ciphertext domain) and IND-CPA security. We think that this result is well-known but could not find a reference which explicitly establishes it.

[31] The reduction works in both the private *and* public key setting since the (IND-)SA adversary calls the encryption oracle in both cases (in order to populate its challenger internal state with well-formed ciphertexts). See Sect. 3.3.1 in [25] (definition of vCCA^D in the public-key case).

Lemma 26. *Let S be a scheme that achieves ciphertext pseudorandomness, then S is IND-CPA secure.*

Proof. Ciphertext pseudorandomness trivially implies SIM-CPA security: under the former property, the ideal experiment in which a trivial simulator replies uniformly picked elements of $\mathcal{C}$ in lieu of ciphertexts is indistinguishable from the real experiment in which genuine ciphertexts are provided. The claim follows from the fact that SIM-CPA is equivalent to IND-CPA [9,64]. □

Proposition 27. *If there exists a scheme S that achieves ciphertext pseudorandomness, then there exists a scheme S' that is IND-CPA secure but does not have ciphertext pseudorandomness.*

Proof. Let S denote a scheme achieving ciphertext pseudorandomness. Consider scheme S' which is built from S by simply padding a parity bit to ciphertexts under S, i.e. let $c = S.\mathsf{Enc}(m)$ then $S'.\mathsf{Enc}(m) = c|\mathsf{pbit}(c)$. Now, because of the parity bit, S' does not achieve ciphertext pseudorandomness. Yet S' is IND-CPA secure by a straightforward reduction to the IND-CPA security of S (which is implied by Lemma 26): the reduction simply transfers all the adversary's encryption and challenge requests as is to the IND-CPA challenger against S and just pads the parity bit to the challenger's replies before forwarding them $\mathcal{A}$. □

F Numerical examples over toy parameters

In this section, we provide toy numerical experiments to help the reader picture the mechanics at work in our various attacks. These have been done with the proof of concept code within the software artifact associated to this paper (script `lincombpoc.py`) that can also run over larger more serious parameters which are provided in the script (see also Sect. F.4 below). So the (toy) parameters that we'll use in the next sections are,

- $q = 2^{32} - 5 = 34359738363$ (has to be prime).
- $\Delta = \lfloor q/2 \rfloor = 17179869181$.
- $n = 4$ (oops!).
- $\sigma_0 = 3.19$.

F.1 Experimenting the attack in the proof of Prop. 11

This section provides a numeric example for the challenge reconstruction attack given in the proof of Prop. 11. We do this against Regev (adversary `Adv1` in script `lincombpoc.py`).

Since $n = 4$, the CPA^D adversary starts by asking its challenger for 5 encryptions of 0,

$$e_0 = [496824929, 3576278474, 3377474290, 1963821999, 1741958112],$$
$$e_1 = [2817233905, 2027409314, 3633330814, 2909846190, 3459522116],$$
$$e_2 = [1868863955, 624401264, 3189557294, 884678600, 3377943814],$$
$$e_3 = [2572654169, 678150079, 2317077602, 3407443867, 2363280825],$$
$$e_4 = [2542450705, 1436789259, 3876104017, 489339899, 173612573].$$

It then asks for a challenge ciphertext over messages 0 and 1 and get in return,

$$\mathsf{ct}^* = [949582366, 2787439785, 2367215987, 1653390094, 3242928605],$$

while the challenger's state now contains the following additional triplet,

$$(0, 1, [949582366, 2787439785, 2367215987, 1653390094, 3242928605]).$$

At this point, an observation of the internals of the challenger that is not available to the adversary reveals that ct^* is an encryption of 1 with noise $e^* = 6$.

The adversary, then does linear algebra on its own to solve linear system,

$$\left(\begin{array}{ccccc|c} 496824929 & 2817233905 & 1868863955 & 2572654169 & 2542450705 & 949582366 \\ 3576278474 & 2027409314 & 624401264 & 678150079 & 1436789259 & 2787439785 \\ 3377474290 & 3633330814 & 3189557294 & 2317077602 & 3876104017 & 2367215987 \\ 1963821999 & 2909846190 & 884678600 & 3407443867 & 489339899 & 1653390094 \\ 1741958112 & 3459522116 & 3377943814 & 2363280825 & 173612573 & 3242928605 \end{array}\right),$$

by means of Gaussian elimination to get,

$$\alpha = [517113441, 1308295495, 3742099423, 1503939484, 4273649667],$$

such that,

$$\mathsf{ct}^* = 517113441e_0 + 1308295495e_1 + 3742099423e_2 + 1503939484e_3 + 4273649667e_4.$$

The adversary then returns to its challenger and uses Lemma 8 and 9 (in the software artifact, functions `extProductUsingChallenger` and `evalLinearCombUsingChallenger` that do not manipulate any ciphertext but only indices from the challenger's internal state) to eventually put a duplication of ct^* in the challenger's internal state. To do so, requires 255 homomorphic additions. At this point, the challenger's state now contains the following additional triplet,

$$(0, 0, [949582366, 2787439785, 2367215987, 1653390094, 3242928605]).$$

Because the left and right cleartext values are both 0 in this latter triplet, the challenger accepts a decryption request from the adversary over its state index. The adversary then receives 1 in return for this request, outputs a guess of 1 of the challenge bit and wins the CPA^D game.

Perhaps, to help better picture the mechanics at work, let us observe the internals of challenger (something which the adversary still cannot do) and look at the term $b-\langle a, \mathsf{sk}\rangle = \Delta m+e$ in the intermediate ciphertexts obtained just before the last summation in Eq. (8) (in the proof of Lemma 9). From the decryption function point of view,

- The first is an enc. of 0 with noise 517113441 i.e. $[\Delta m + e]_q = 517113441$.
- The second is an enc. of 1 with noise -469107344 i.e. $[\Delta m+e]_q = 16710761837$.
- The third is an enc. of 1 with noise 616855695 i.e. $[\Delta m + e]_q = 17796724876$.
- The fourth is an enc. of 1 with noise -643544161 i.e. $[\Delta m+e]_q = 16536325020$.
- The fifth is an enc. of 0 with noise -21317624 i.e. $[\Delta m + e]_q = 34338420739$.

Then,

$$[517113441+16710761837+17796724876+16536325020+34338420739]_q = 17179869187 \tag{34}$$

which is exactly $\Delta + 6$ ($\Delta = 17179869181$) and thus consistent with the fact (unknown to the adversary) that ct^* is an encryption of 1 with noise 6.

In some sense, the values in (34) form shares of $\Delta m_\gamma + e^*$ that the adversary has obtained obliviously (as it worked only over the ciphertext domain).

F.2 Experimenting the reduction in the proof of Prop. 13

This section provides a numeric example for the reduction in the proof of Prop. 13. We attack the Regev scheme by being able to obtain the decryption of an ill-formed ciphertext,

$$[[-\Delta]_q, 0, ..., 0; 0],$$

that leaks the first bit of sk (see also Sect. D). So we illustrate how to turn a CCA1 attack against Regev into another attack which is valid in the CPA^D model (and even in the KR^D model since bits of sk are obtained by the adversary). This is adversary `Adv2` in script `lincombpoc.py`.

First, an observation of the internals of the challenger that is not available to the adversary reveals that the first bit of sk is 0.

Since $n = 4$, the CPA^D adversary starts by asking its challenger for 5 encryptions of 0,

$$e_0 = [692006638, 3249315990, 3022178557, 3807840622, 3807840613],$$
$$e_1 = [3667791361, 832030391, 2923726872, 4061795943, 4061795943],$$
$$e_2 = [3040272744, 1603024470, 2105542969, 2966284021, 2966284019],$$
$$e_3 = [719513982, 223409924, 3106909093, 603931678, 603931681],$$
$$e_4 = [1656639534, 2741788126, 305246404, 2029906909, 2029906905].$$

The adversary, then does linear algebra on its own to solve linear system,

$$\left(\begin{array}{ccccc|c} 692006638 & 3667791361 & 3040272744 & 719513982 & 1656639534 & 2147483646 \\ 3249315990 & 832030391 & 1603024470 & 223409924 & 2741788126 & 0 \\ 3022178557 & 2923726872 & 2105542969 & 3106909093 & 305246404 & 0 \\ 3807840622 & 4061795943 & 2966284021 & 603931678 & 2029906909 & 0 \\ 3807840613 & 4061795943 & 2966284019 & 603931681 & 2029906905 & 0 \end{array}\right),$$

by means of Gaussian elimination to get,

$$\alpha = [1521485173, 3701247994, 1637185992, 1718881501, 3489677427].$$

The adversary then returns to its challenger and uses Lemma 8 and 9 (in the software artifact, functions `extProductUsingChallenger` and `evalLinearCombUsingChallenger` that do not manipulate any ciphertext but only indices from the challenger's internal state) to eventually put its target ciphertext in the challenger's internal state. To do so, requires 243 homomorphic additions. At this point, the challenger's state now contains the following additional triplet,

$$(0, 0, [2147483646, 0, 0, 0, 0])$$

Because the left and right cleartext values are both 0 in this latter triplet, the challenger accepts a decryption request from the adversary over this state index. The adversary then receives 0 in return for this request, outputs a guess of 0 for the first bit of sk, which is correct.

F.3 Attack of Sect. 5.3 against DE-CKKS

In this section, we illustrates the mechanics at work when turning the CCA1 attack against DE-CKKS that we present in Sect. 5.3 into one that fits the CPA^D model. Per se, it is another illustration of the reduction in the proof of Prop. 13. In this section, we will consider a single noise slot (termed imaginary slots in DE-CKKS/OpenFHE terminology) of a ciphertext under DE-CKKS. This is equivalent to working over a message-less variant of Regev.

This is adversary `Adv4` in script `lincombpoc.py`.

Like previously, since $n = 4$, the CPA^D adversary starts by asking its challenger for 5 encryptions of 0,

$$e_0 = [375795092, 226225790, 1085877351, 1000565249, 1000565250],$$
$$e_1 = [585836817, 2790773064, 753512806, 4034082287, 4034082286],$$
$$e_2 = [2587322279, 2321693833, 3281071001, 2991453698, 2991453706],$$
$$e_3 = [831937000, 3784854820, 3577885966, 636282055, 636282052],$$
$$e_4 = [2018939840, 812292333, 3370983633, 1049721613, 1049721613].$$

Its then asks for another ciphertext to work with,

$$\mathsf{ct} = [1790315768, 1648888279, 18010583, 3254000877, 3254000875],$$

while the challenger's state now contains the following additional triplet,

$$(0, 0, [1790315768, 1648888279, 18010583, 3254000877, 3254000875]).$$

At this point, an observation of the internals of the challenger that is not available to the adversary reveals that ct's noise is -2.

Then the adversary modifies its own local copy of ct by adding 3 to the b terms of that ciphertext eventually getting, via this non-legitimate operation, a ciphertext which noise is 1. Unfortunately for the adversary, this modified ciphertext is of course not in the challenger's state since it was "actively" obtained by direct manipulation of the noise slot.

The adversary then targets this modified ciphertext,

$$\mathsf{ct}' = [1790315768, 1648888279, 18010583, 3254000877, 3254000878],$$

and does linear algebra on its own to solve linear system,

$$\left(\begin{array}{ccccc|c} 375795092 & 585836817 & 2587322279 & 831937000 & 2018939840 & 1790315768 \\ 226225790 & 2790773064 & 2321693833 & 3784854820 & 812292333 & 1648888279 \\ 1085877351 & 753512806 & 3281071001 & 3577885966 & 3370983633 & 18010583 \\ 1000565249 & 4034082287 & 2991453698 & 636282055 & 1049721613 & 3254000877 \\ 1000565250 & 4034082286 & 2991453706 & 636282052 & 1049721613 & 3254000878 \end{array}\right),$$

by means of Gaussian elimination to get,

$$\alpha = [2039122872, 1684505479, 642691365, 3263705201, 1283945062],$$

such that,

$$\mathsf{ct}' = 2039122872e_0 + 1684505479e_1 + 642691365e_2 + 3263705201e_3 + 1283945062e_4.$$

The adversary then returns to its challenger and uses Lemma 8 and 9 (in the software artifact, functions `extProductUsingChallenger` and `evalLinearCombUsingChallenger` that do not manipulate any ciphertext but only indices from the challenger's internal state) to eventually put ct' in the challenger's internal state. To do so, requires 240 homomorphic additions. At this point, the challenger's state now contains the following additional triplet,

$$(0, 0, [1790315768, 1648888279, 18010583, 3254000877, 3254000878]).$$

Because the left and right cleartext values are both 0 in this latter triplet, the challenger accepts a decryption request from the adversary over its state index. When decrypting, the decryption function gets a noise of 1 in this slot (that is then eventually used for the decryption of other slots).

Again, to help better picture the mechanics at work, let us observe the internals of challenger (something which the adversary still cannot do) and look at the noises in the intermediate ciphertexts obtained just before the last summation in Eq. (8) (in the proof of Lemma 9),

- The first has noise 2039122872.
- The second has noise -1684505479.
- The third has noise 846563629.
- The fourth has noise -1201181021.
- The fifth has noise 0.

Then, although most of them have a very large noise, they collectively sum to 1 modulo q,

```
>>> (2039122872-1684505479+846563629-1201181021+0)%(2**35-5)
1
```

which is exactly the noise in the original ciphertext (-2) plus 3.

In some sense, the values above form shares of the target noise that the adversary has obtained obliviously (as $\mathcal{A}$ worked only over the ciphertext domain).

F.4 Experimenting with "more serious" parameters

For completeness, we have also run the above experiments over more serious parameters that achieves around 128 bits of security according to the lattice-estimator. These parameters are:

- $q = 2^{32} - 5$.
- $n = 900$.
- $\sigma_0 = 2^{17}$.
- Binary secret.

Since all the CPA^D adversaries in the `lincombpoc.py` script solve only one linear system and perform one ciphertext reconstruction, they all have approximately the same execution time. On average, running one adversary took 94.94 secs, with 86.78 secs (91%) spent on the Gaussian elimination and 6.64 secs (6.99%) spent on ciphertext reconstruction. Recall that these figures have been obtained with a non-optimized Python implementation so they can easily be improved by several orders of magnitude.

Table of Contents

Impossibility of CPA^D security for a class of FHE schemes 1
Marina Checri, Pierre-Emmanuel Clet, Marc Renard and Renaud Sirdey

1 Introduction 1
1.1 Overview of our contributions 3
1.2 Paper organization 5
2 Preliminaries 5
2.1 Correctness properties 6
2.2 Security notions 7
CPA^D security (and KR^D, CPA$_1^D$, CPA$_2^D$ security). 7
Application-aware Homomorphic Encryption. 9
vCCA security (and others). 9
2.3 The basic Regev Scheme ($\mathcal{R}$) 10
2.4 Smudging 11
3 HELLHO schemes: definitions and first properties 12
3.1 HELLHO and $\mathbb{F}$-HELLHO schemes 12
3.2 Concrete HELLHO schemes 15
4 Impossibility results 17
4.1 On the existence of CPA^D-secure HELLHO 17
4.2 On the existence of $\mathcal{F}(\lambda)$-correct HELLHO schemes 19
4.3 Relation between CCA1 and CPA$_1^D$ for HELLHO schemes 19
4.4 Impossibility of (IND-)SA, IV-CCA, vCCA and vCCA^D security for HELLHO schemes 20
5 Consequences for (R)LWE-based schemes with Dynamic Error Estimation (DEE) 22
5.1 CPA^D security and the DEE heuristic 23
5.2 KR^D (in)security of DE-RLWE 25
5.3 KR^D (in)security of DE-CKKS 26
6 Beyond HELLHO schemes? 29
7 Responsible disclosure 29
A Appendix sections organization 34
B Additional lemmas and proofs 34
B.1 Additional lemmas 34
B.2 Alternative (direct) proof of Prop. 12 35
B.3 CPA^D security with a "variance oracle" 36
CPA^D security via ciphertext-independent noise variance monitoring. 38
C Additional details on our KR^D attacks against DE-RLWE and DE-CKKS 39
C.1 Improved complexity analysis for the attacks in Sect. 5.2 and 5.3 39
DE-RLWE (Sect. 5.2). 39
DE-CKKS (Sect. 5.2). 39

C.2 Polynomial fitting procedure in the attack of Sect. 5.3 40
C.3 Porting the attack of Sect. 5.3 to DE-CKKS with encoding (sketch) 42
D Folklore CCA1 attacks against Regev 44
D.1 A non-adaptive "ill-formed" ciphertext-based attack 44
D.2 An adaptive noise recovery attack 44
E Additional details on security notions 44
E.1 CPA, CCA1 and CCA2 security. 44
E.2 Application-aware Homomorphic Encryption(AAHE) 45
E.3 vCCA^D security 46
E.4 (IND-)SA security 47
E.5 Ciphertext pseudorandomness vs IND-CPA 48
F Numerical examples over toy parameters 49
F.1 Experimenting the attack in the proof of Prop. 11 49
F.2 Experimenting the reduction in the proof of Prop. 13 51
F.3 Attack of Sect. 5.3 against DE-CKKS 52
F.4 Experimenting with "more serious" parameters 54