IACR Communications in Cryptology
ISSN 3006-5496, Vol. 2, No. 1, 35 pages.

https://doi.org/10.62056/aee0iv7sf

Relations Among New CCA Security Notions for Approximate FHE

Chris Brzuska[1], Sébastien Canard[2], Caroline Fontaine[3], Duong Hieu Phan[2], David Pointcheval[4,5], Marc Renard[3,6] and Renaud Sirdey[6]

[1] Aalto University, Espoo, Finland
[2] Télécom Paris, Institut Polytechnique de Paris, Palaiseau, France
[3] Université Paris-Saclay, CNRS, ENS Paris-Saclay, Laboratoire Méthodes Formelles, Gif-sur-Yvette, France
[4] DIENS, Ecole normale supérieure, CNRS, Inria, PSL University, Paris, France
[5] Cosmian, Paris, France
[6] Université Paris-Saclay, CEA, List, Palaiseau, France

Abstract.
In a recent Eurocrypt'24 paper, Manulis and Nguyen have proposed a new CCA security notion, vCCA, and associated construction blueprints to leverage both CPA-secure and correct FHE beyond the CCA1 security barrier. However, because their approach is only valid under the correctness assumption, it leaves a large part of the FHE spectrum uncovered, as many FHE schemes used in practice turn out to be approximate and, as such, do not satisfy the correctness assumption. In this paper, we improve their work by defining and investigating a variant of their security notion which is suitable for a more general case where approximate FHE are included. As the passive security of approximate FHE schemes is more appropriately captured by CPA^D rather than CPA security, we start from the former notion to define our vCCA^D new security notion. Although we show that vCCA and vCCA^D are equivalent when the correctness assumption holds, we establish that vCCA^D security is strictly stronger than vCCA security in the general case. In doing so, we interestingly establish several new separation results between variants of CPA^D security of increasing strength. This allows us to clarify the relationship between vCCA security and CPA^D security, and to reveal that the security notions landscape is much simpler for correct FHE than when approximate ones are included — in which case, for example, we establish that multiple challenges security notions are strictly stronger than single-challenge ones for both CPA^D and vCCA^D security. Lastly, we also give concrete construction blueprints, showing how to leverage some of the blueprints proposed by Manulis and Nguyen to achieve vCCA^D security. As a result, vCCA^D security is the strongest CCA security notion known so far to be achievable by both correct and approximate FHE schemes.

Keywords: FHE · CPA^D · CCA security · SNARK · Verifiability

1 Introduction

Since its inception more than ten years ago, Fully Homomorphic Encryption (FHE) has been the subject of a lot of research toward more efficiency and better practicality. From a security

This work was supported by the France 2030 ANR Projects ANR-22-PECY-003 SecureCompute.
E-mail: chris.brzuska@aalto.fi (Chris Brzuska), sebastien.canard@telecom-paris.fr (Sébastien Canard), caroline.fontaine@cnrs.fr (Caroline Fontaine), hieu.phan@telecom-paris.fr (Duong Hieu Phan), david.pointcheval@ens.fr (David Pointcheval), marc.renard@cea.fr (Marc Renard), renaud.sirdey@cea.fr (Renaud Sirdey)

Received: 2025-01-10 Accepted: 2025-03-11

perspective, however, FHE still raises a number of questions and challenges. In particular, all the FHE usable in practice, BFV [Bra12, FV12], BGV [BGV12], CKKS [CKKS17] and TFHE [CGGI16], achieve CPA security but are trivially CCA1 insecure. Although it is well-known that malleability is contradictory with CCA2 security, building efficient FHE constructions achieving some degree of CCA security (e.g. CCA1) remains a very important open challenge.

From a theoretical perspective, a significant step has been recently achieved by Manulis and Nguyen in [MN24] with the introduction of the notion of vCCA security, which is proven to be strictly stronger than CCA1 security while being achievable by FHE-based malleable schemes through several construction blueprints. In essence, these construction strategies consist in starting from a CPA secure and *correct* FHE and augmenting it with the machinery required for proving the well-formedness of *fresh* ciphertexts (i.e. ciphertexts which are direct outputs of the encryption function) as well as that of *evaluated* ciphertexts (i.e. ciphertexts derived from well-formed fresh ciphertexts by means of genuine homomorphic operations), with the decryption function of the augmented scheme returning $\perp$ when the proof verification fails. The intuition behind such construction strategies is that the proof machinery downgrades attackers to CPA ones and that, as a result, some form of CCA security is achieved by the augmented scheme. Although several techniques can be used to ensure the well-formedness of fresh ciphertexts (such as signatures in the private key setting, or Naor-Yung [NY90] in the public key setting), their approach intimately relies on Succinct Non-interactive Arguments of Knowledge (SNARKs) to enforce the well-formedness of evaluated ciphertexts. Under the assumption that the underlying SNARK is *simulation-extractable*, it then becomes possible to define a new (single challenge) security game along with a new security notion called vCCA, in the spirit of the CCA2 game: whereas the (second step) CCA2 game decryption oracle rejects the challenge ciphertext, the vCCA security game (second step) decryption oracle rejects all byproducts of the challenge ciphertext, identified by means of the SNARK extractor. One of Manulis and Nguyen's main contributions is to show, interestingly, that the resulting security notion, vCCA, is strictly stronger than CCA1 (no second step oracle) as well as a strict relaxation of CCA2 (because the CCA2 decryption oracle is more permissive than the vCCA decryption one). They also investigate the relationship between vCCA security and other CCA2 relaxations such as RCCA, and others.

Another very important security notion for FHE, introduced by Li and Micciancio in [LM21], is that of CPA^D security. It formalizes the security of FHE against a slight and seemingly benign extension of CPA security: in CPA^D, the adversary is granted access only to a highly constrained decryption oracle which accepts only genuine ciphertexts, or ciphertexts derived from genuine ciphertexts by means of genuine homomorphic operations. The initial intuition is that, by knowing the cleartext inputs of an FHE calculation, the adversary should be able to compute all the outputs of that decryption oracle by his or herself and, as a consequence, that CPA^D security is implied by or even equivalent to CPA security. However, the correctness assumption implicitly lies at the heart of this reasoning and Li and Micciancio [LM21] demonstrated that these intuitions are not true for approximate FHE schemes such as CKKS for which it turns out that the CPA^D decryption oracle outputs leak the LWE noises in the ciphertexts, resulting in the ability for the adversary to easily and practically recover the secret decryption key of the scheme. Although initially introduced for approximate FHE, recent works [CSBB24, CCP+24] have shown that the non-approximate FHE schemes that were previously considered immune to CPA^D attacks are, contrary to this folklore belief, all CPA^D insecure as soon as decryption errors can (or can be made to) occur with a non-negligible probability.

In their paper, Manulis and Nguyen [MN24] define and study vCCA security only under the correctness assumption and address only very briefly CPA^D security, essentially claiming informally that their vCCA scheme construction blueprints also apply to approximate

FHE "with the caveat that approximate FHE schemes need to be CPA^D-secure". In the present paper, we clarify the relationship between vCCA security and CPA^D security, and propose a new CCA security notion, vCCA^D, covering the spectrum of both correct *and* approximate FHE schemes. We show that both notions are equivalent when the correctness assumption holds, but establish that vCCA^D security is strictly stronger than vCCA security in the general case where approximate FHE are allowed. In doing so, we interestingly establish several new separation results between variants of CPA^D security of increasing strength. This allows us to show that vCCA security does not imply CPA^D security, but rather a much weaker single-challenge "CCA1 style" variant of it. We also reveal that the security notion landscape is much simpler for correct FHE than in the general case where, for example, we establish that multiple challenges security notions are strictly stronger than single-challenge ones for both CPA^D and vCCA^D security. Lastly, we study concrete construction blueprints, and show how to leverage some of the blueprints proposed in [MN24] to achieve the new and stronger vCCA^D security.

Our first motivation for this study is practical: although the construction blueprints discussed in [MN24] and Sec. 6 may not be amenable to efficient implementations in their full generality, it seems possible to instantiate them in use-cases requiring only special classes of homomorphic calculations to be performed. When this is possible, vCCA and vCCA^D security provide a yardstick to achieve CCA security in a context where decryption oracles often naturally occur in many application scenarios for FHE. We elaborate more on this point in our concluding remarks (Sec. 7). Our second motivation is more theoretical, as this line of works allows to better understand and probe the limitations of FHE in terms of CCA security with, as we establish in this paper, vCCA^D security as the strongest CCA security notion known so far to be achievable by FHE in the general regime where approximate/non-exact schemes are considered (with most practically used FHE schemes falling in that later category).

1.1 Summary of security notions and contributions

In this work, we study the following (non standard) security notions:

- CPA^D: the multiple challenges passive security notion introduced in [LM21] for approximate FHE.
- $\mathrm{CPA}_2^D \equiv \mathrm{CPA}_{\mathrm{SC}}^D$: restriction of CPA^D to the single challenge case.
- CPA_1^D: restriction of CPA_2^D with the decryption oracle closing after the challenge request (similar in spirit to the CCA1/CCA2 definitions). Note that CPA_1^D is different from non-adaptive CPA^D as defined and studied in [LM21].
- $\mathrm{vCCA}_{\mathrm{SC}}$: the single challenge CCA security notion introduced in [MN24] for correct FHE. Note that it is simply denoted vCCA in [MN24].
- vCCA: the multiple challenges variant of vCCA (this variant was not considered in [MN24]).
- vCCA^D: our main new multiple challenges CCA security notion for FHE in the general regime which includes approximate FHE.
- $\mathrm{vCCA}_{\mathrm{SC}}^D$: restriction of vCCA^D to the single challenge case.

Note that, following standard conventions from [BDJR97], the multiple challenges notions will sometimes be prefixed by LOR- to avoid any ambiguity with the corresponding single challenge notions (also known as FTG).

With this in mind, the contributions of this paper are the following:

- When the correctness assumption holds for the underlying FHE, we show that:
  - CPA_1^D, $\text{CPA}_2^D \equiv \text{CPA}_{\text{SC}}^D$ and CPA^D security are all equivalent to CPA security.
  - vCCA_{SC}, vCCA, $\text{vCCA}_{\text{SC}}^D$ and vCCA^D security are all equivalent.
- In the general case where the correctness assumption does not necessarily hold and approximate FHE are allowed, the picture we reveal is much more interesting:
  - For CPA^D security, we establish that $\text{CPA}_1^D \prec \text{CPA}_2^D \prec \text{CPA}^D$, where the $\prec$ operator stands for the "is strictly weaker" relation. As a bonus, this settles the question of the relationship between single and multiple-challenge CPA^D security that was left open in [LM21].
  - We clarify the relationship between vCCA and CPA^D security by showing that $\text{CPA}_1^D \prec \text{vCCA}_{\text{SC}}$ but that vCCA_{SC} security implies neither CPA_2^D nor CPA^D security (and vice-versa), contrary to what was informally claimed in [MN24].
  - For vCCA security, we further establish that $\text{vCCA}_{\text{SC}} \equiv$ vCCA and that $\text{vCCA}_{\text{SC}} \prec \text{vCCA}_{\text{SC}}^D$. Thus, we demonstrate that our new security notion is strictly stronger than vCCA security, even in the single challenge case.
  - Lastly, for vCCA^D security, we prove that $\text{CPA}^D \prec \text{vCCA}^D$, and further prove that $\text{vCCA}_{\text{SC}}^D \prec \text{vCCA}^D$. This implies that vCCA^D is the strongest of all these notions in the general FHE case and that it is the one that should be strived for.
- Lastly, we revisit the CPA-to-vCCA FHE scheme construction blueprints proposed in [MN24] under the correctness assumption and turn them, when possible, into CPA^D-to-vCCA^D blueprints. In particular, we are able to do so and prove the vCCA^D security of the Encrypt-then-MAC (private key, designated verifier), Encrypt-then-Sign (private key, public verifier), and CCA2-Companion-Ciphertext (public key, designated verifier) blueprints; but we also show that the Naor-Yung-based blueprint (public key, public verifier) cannot be used when the (perfect) correctness assumption does not hold. In this later case (public key, public verifier), we thus propose a new blueprint, Encrypt-then-Prove, which achieves vCCA^D (and vCCA) security in the general setting where the correctness assumption does not necessarily hold.

Figure 1 summarizes the relationships between these notions in the general case. When proving relationships between security notions, we make a difference between the *correct case* or *correct regime*, where the FHE correctness assumption is assumed to hold, and the *general case* or *general regime*, where approximate FHE are allowed. This terminology is used consistently in the paper.

1.2 Paper organization

This paper is organized as follows. After some preliminaries (Sect. 2), Sect. 3 introduces the definition of vCCA^D security and recalls the definitions of CPA^D and vCCA security. In this section, we also investigate the definitional connections between these notions. Then, in Sect. 4, we investigate the relationship among the single challenge variants of these security notions in both the restricted setting where the correctness assumption holds and in the general case where approximate FHE are allowed. Then, in Sect. 5, we focus on unveiling the relationships between the single and multiple-challenge variants of these security notions as well as the relationships between the multiple-challenge variants between each other. Sect. 6 is devoted to prove the security of several generic scheme construction blueprints with respect to the stronger multiple-challenge vCCA^D security notion. Sect. 7 then concludes the paper.

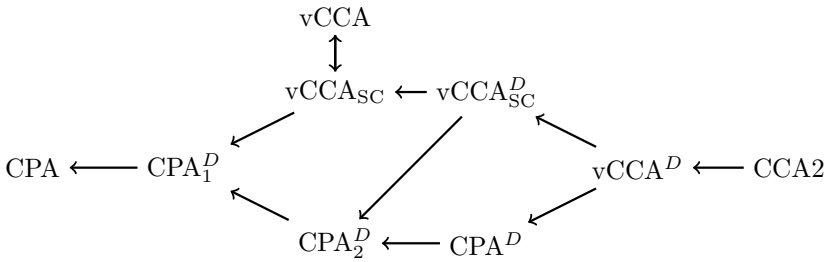

Figure 1: Summary of the security notions investigated in this paper and their relationships in the general regime, where approximate FHE are allowed. Note that all single arrows are strict implications, and recall that $\mathrm{CPA}^D_2 \equiv \mathrm{CPA}^D_{\mathrm{SC}}$ is the restriction of CPA^D to the single challenge case and that CPA^D_1 is the restriction of CPA^D_2 with the decryption oracle closing after the challenge request. Please also remind that notions without a subscript in their names are multiple challenge ones and that, for consistency, we denote by $\mathrm{vCCA}_{\mathrm{SC}}$ the single challenge security notion defined in [MN24]. We emphasize that all the relations shown on this figure involve at least one new notion introduced in this paper and are, as a consequence, new relations.

2 Preliminaries

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 $\mathcal{C}$ as a triplet of PPT algorithms:

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

Let COIN denote the randomness space of $\mathcal{E}$. We will sometimes externalize the randomness used in the encryption function by means of the notation $\mathsf{Enc}_{\mathsf{ek}}(m; r)$, with $m \in \mathcal{P}$ and $r \in \mathsf{COIN}$ (in this case, the function $\mathsf{Enc}_{\mathsf{ek}} : \mathcal{P} \times \mathsf{COIN} \longrightarrow \mathcal{C}$ is deterministic). When ek is public, we say that $\mathcal{E}$ is a *public-key* encryption scheme. Conversely, when ek has to remain private and an adversary can create valid ciphertexts with at most $\mathrm{neg}(\lambda)$ probability using only public knowledge, we say that $\mathcal{E}$ is a *private-key* encryption scheme. When for all $(\mathsf{ek}, \mathsf{sk}) \in \mathcal{K}$ and all $m \in \mathcal{P}$ we have that

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

we say that $\mathcal{E}$ is *correct*. When the above equality always holds, we will talk about *perfect* correctness. A ciphertext is *valid* if it is the output of the encryption function for some message $m \in \mathcal{P}$, that is, if there exists $m \in \mathcal{P}$ and some randomness $r \in \mathsf{COIN}$ such that $c = \mathsf{Enc}_{\mathsf{ek}}(m; r)$. We further say that $\mathcal{E}$ is *verifiable* if there exists a PPT algorithm Verif, taking a ciphertext as input, which tells whether or not this ciphertext is valid with a $\mathrm{neg}(\lambda)$ probability of error for an adversary with knowledge of only public data. For verifiable schemes, the decryption function with input ciphertext c outputs $\perp$ when $\mathsf{Verif}(c) = \mathsf{False}$. Note that for all non-homomorphic schemes considered in this paper, $\mathsf{Dec}_{\mathsf{sk}}$ will always be a *deterministic* polynomial-time algorithm. When there is no ambiguity, we omit the ek and sk subscripts to Enc and Dec to lighten the notation.

Given a function class $\mathcal{F}_H$, we define a homomorphic encryption (HE) scheme $\mathcal{E}_H$ as an encryption scheme augmented by a *deterministic*[1] polynomial-time algorithm Eval which,

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

on input $f \in \mathcal{F}_H$ and $c_1, ..., c_K \in \mathcal{C}^K$, where K denotes the arity of function f, outputs a new *evaluated* ciphertext. When $\mathcal{E}_H$ satisfies condition (1) and when Eval is such that for all $(\mathsf{ek}, \mathsf{sk}) \in \mathcal{K}$, all $f \in \mathcal{F}_H$ and all $m_1, ..., m_K \in \mathcal{P}^K$

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

we say that $\mathcal{E}_H$ is a *correct* HE scheme. When this is not the case, we say that $\mathcal{E}_H$ is an *approximate* HE scheme. Consistently with [LMSS22], 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

$$\Pr_{\vec{r} \in \mathsf{COIN}^K}(||\mathsf{Dec}(\mathsf{Eval}(f, \mathsf{Enc}(m_1; r_1), ..., \mathsf{Enc}(m_K; r_K))) - f(m_1, ..., m_K)||_\infty \leq \varepsilon) \geq p_\varepsilon, \quad (2)$$

with $p_\varepsilon \geq \frac{3}{4}$[2]. We will sometimes refer to a scheme satisfying this property as an ε-correct scheme. When $\mathcal{E}_H$ achieves correctness only for $\mathcal{F}_C \subset \mathcal{F}_H$, it is said to be $\mathcal{F}_C$-*correct* (in the spirit of [ABMP24]).

All the HE schemes we consider in this paper are public-key. Also note that for the homomorphic schemes considered in this paper, $\mathsf{Dec}_{\mathsf{sk}}$ is by default a *deterministic* polynomial-time algorithm, unless explicitly stated otherwise (e.g. CKKS with noise flooding as defined in [LMSS22] has a probabilistic decryption algorithm). When $\perp \in \mathcal{E}_H.\mathcal{P}$, we will further always assume a *consistency* property which requires that $\forall \vec{m}, \vec{r} \in \mathcal{P}^K \times \mathsf{COIN}^K$,

$$\mathsf{Dec}(\mathsf{Eval}(f, \mathsf{Enc}(m_1; r_1), ..., \mathsf{Enc}(m_K; r_K))) \neq \perp, \quad (3)$$

and

$$\mathsf{Dec}(\mathsf{Eval}(f, c_1, ..., c_K)) = \perp, \quad (4)$$

whenever $\exists i : \mathsf{Dec}(c_i) = \perp$.

As in [MN24], for signatures and MAC we use the standard definitions respectively $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ and $M = (\mathsf{KeyGen}, \mathsf{Tag}, \mathsf{Verify})$ and assume SUF-CMA security. In our case, EUF-CMA security will not be sufficient because we essentially use signatures as a building block for CCA2 encryption schemes in the private key setting. Lastly, we consider straightline-extractable SNARK, $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$, over function class $\mathcal{F}_E$ (slightly departing from [MN24] which required simulation-extractability). Indeed, in the security proofs of the constructions we study in Sect. 6, we are in the setup where only the adversary generates proofs and where our simulator only invokes the SNARK extractor when $\Pi.\mathsf{Verify}(\pi) = \mathsf{True}$ on a proof π. Additionally, because we do not investigate circuit privacy, we do not need zk-SNARKs to prove correct homomorphic derivations. Formal definitions for these notions are recalled in appendix A.

3 Defining vCCA^D security

This section introduces our vCCA^D security notion which is an extension of both vCCA and CPA^D (hence the name). As they both are extensively referred to in this paper, we start by recalling the games associated with these two notions. We then define the vCCA^D game.

3.1 The CPA^D game

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

[2]We use this probability only to prove non-negligible advantages in some of our proofs. In practice p_ε is typically chosen above $1 - 2^{-40}$. In some contexts, e.g. [ABMP24], p_ε even has to be at least $1 - \mathrm{neg}(\lambda)$.

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 $b \xleftarrow{\$} \{0,1\}$, unknown to $\mathcal{A}$, and an initially empty state S of message-message-ciphertext triplets:

- Key generation. Run $(\mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and give ek to $\mathcal{A}$ (when the scheme is public-key).
- Encryption request. When $\mathcal{A}$ queries $(\texttt{plaintext}, m)$, $m \in \mathcal{P}$ compute $c = \mathsf{Enc}(m)$, give c to $\mathcal{A}$ and do
$$S := [S; (m, m, c)].$$
- Challenge request. When $\mathcal{A}$ queries $(\texttt{test messages}, m_0, m_1)$, $m_0, m_1 \in \mathcal{P}^2$ $(m_0 \neq m_1)$ compute $c = \mathsf{Enc}(m_b)$, give c to $\mathcal{A}$ and do
$$S := [S; (m_0, m_1, c)].$$
- Evaluation request. When $\mathcal{A}$ queries $(\texttt{eval}, f, l_1, \ldots, l_K)$ $(l_i \in \{1, ..., |S|\}, \forall i)$, compute
$$m'_0 = f(S[l_1].m_0, \ldots, S[l_K].m_0),$$
and
$$m'_1 = f(S[l_1].m_1, \ldots, S[l_K].m_1),$$
as well as
$$c' = \mathsf{Eval}(f, S[l_1].c, \ldots, S[l_K].c),$$
give c' to $\mathcal{A}$ and do
$$S := [S; (m'_0, m'_1, c')].$$
- Decryption request. When $\mathcal{A}$ queries $(\texttt{ciphertext}, l)$ $(l \in \{1, ..., |S|\})$ proceed as follows: if $S[l].m_0 \neq S[l].m_1$ then return $\perp$ to $\mathcal{A}$. Otherwise return $\mathsf{Dec}(S[l].c)$.
- Guessing stage (after polynomially many interleaved encryption and decryption requests). When $\mathcal{A}$ outputs $(\texttt{guess}, b')$, the outcome of the game is determined as follows. If $b' = b$ then $\mathcal{A}$ wins the game. Otherwise, $\mathcal{A}$ loses the game.

A number of points should be emphasized with respect to the above game. First, the decryption oracle accepts only ciphertexts from the game state which are necessarily well-formed (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 well-formed ciphertexts). As such, the above game does not capture any CCA aspects. Second, when $S[l].m_0 = S[l].m_1$, it is important that the decryption oracle returns $\mathsf{Dec}(S[l].c)$ and not $S[l].m_0$ (or, equivalently in that case, $S[l].m_1$). For correct FHE, this has no impact, as $\mathsf{Dec}(S[l].c) = S[l].m_0 = S[l].m_1$ in that case (and, as $\mathcal{A}$ learns nothing it does not already know, CPA^D security coincides with CPA security for correct FHE). However, for approximate FHE, even when $S[l].m_0 = S[l].m_1$, we have (with overwhelming probability) that $\mathsf{Dec}(S[l].c) \neq S[l].m_0$ and $\mathsf{Dec}(S[l].c) \neq S[l].m_1$. Thus for approximate FHE, the decryption oracle grants $\mathcal{A}$ access to information she cannot compute on her own, possibly resulting in a guessing advantage depending on whether or not the cryptosystem at hand is CPA^D secure. Additionally, let us also emphasize that, in the above game, $\mathcal{A}$ has control of the homomorphic calculations that are performed as f is included in the evaluation request. As a last remark, we acknowledge the fact that

explicitly adding encryption requests to the above game is redundant as these are simply challenge requests with $m_0 = m_1$ (as was assumed in the original definition of [LM21]). However, since we are going back and forth between single and multiple challenges security notions in the sequel, we feel that this explicitation may avoid later confusions. Although the number of allowed challenge requests may vary from one security notion to another, the number of encryption requests an adversary can perform always remains "unlimited".

In addition to the above multiple-challenge notion initially defined in [LM21], we also define and consider the following weaker restrictions of it:

- $\mathrm{CPA}_2^D \equiv \mathrm{CPA}_{\mathrm{SC}}^D$: restriction of CPA^D to the single challenge case where the adversary is allowed only one request of the form $(\texttt{test messages}, m_0, m_1)$ with $m_0 \neq m_1$.
- CPA_1^D: restriction of CPA_2^D with the decryption oracle closing after the challenge request (similar in spirit to the CCA1/CCA2 definitions, hence the choice for the notations).

Note that CPA_1^D is different from non-adaptive CPA^D as defined and studied in [LM21]. Indeed, there is a slight difference between the notion of adaptability as understood in the multiple-challenge context of [LM21] (the adversary performs all its requests at once) 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).

3.2 The vCCA game

We now consider the vCCA game as recently introduced in [MN24]. Contrary to the CPA^D game presented in the previous section, that game is *single challenge* meaning that the adversary performs only one request to the challenge oracle with $m_0 \neq m_1$. As already stated in Sect. 1.1, we will now consistently refer to the original game and security notion in [MN24] as $\mathrm{vCCA}_{\mathrm{SC}}$ and reserve the vCCA naming for its multiple challenge generalisation which we introduce and study in Sect. 5. Also, in the $\mathrm{vCCA}_{\mathrm{SC}}$ security game, the cryptosystem is augmented with a PPT witness extractor $\mathsf{Extract} : \mathcal{C} \times \mathcal{X} \longrightarrow \mathcal{F}_E \cup \{\mathsf{id}\} \times \mathcal{C}^*$, where $\mathcal{X}$ denotes a set of auxiliary data, such that:

- For any ciphertext $c \in \mathcal{C}$ which is obtained by invoking $\mathsf{Eval}(f, c_1, ..., c_K)$,

$$\mathsf{Extract}(c, \mathsf{aux}) = (f, c_1, ..., c_K).$$

- Otherwise, $\mathsf{Extract}(c, \mathsf{aux}) = (\mathsf{id}, c)$.

Let us emphasize that all the construction blueprints which will be discussed in Sect. 6 embed proof material in their ciphertexts and rely on a SNARK to enforce correct homomorphic evaluations over some input ciphertexts, Intuitively, the above $\mathsf{Extract}$ thus corresponds to the extractor of that underlying SNARK which allows to retrieve a witness from the proven statement as well as auxiliary data forming the trace of the execution of the adversary (as formally defined in Sect. A). Because such an extractor exists only with respect to provers able to produce valid proofs, we will further assume, in the case of Designated-Verifier SNARKs, that the adversary is given access to a verification oracle. By abuse of notations we will omit the aux argument in the sequel.

In particular, the above definition implies that

$$\mathsf{Extract}(\mathsf{Eval}(f, \mathsf{Enc}(m_1, r_1), ..., \mathsf{Enc}(m_K, r_K))) = (f, \mathsf{Enc}(m_1, r_1), ..., \mathsf{Enc}(m_K, r_K)). \quad (5)$$

By extension, we also expect that

$$\mathsf{Extract}(\mathsf{Enc}(m, r)) = (\mathsf{id}, \mathsf{Enc}(m, r)). \quad (6)$$

Being single challenge, the vCCA_{SC} game therefore has two decryption oracles[3]. Before the unique challenge encryption oracle request, the first step decryption oracle is simply as follows.

- Decryption request (1st step). When $\mathcal{A}$ queries $(\texttt{ciphertext}, c)$: return her $\mathsf{Dec}(c)$.

Then, after the generation of the unique challenge ciphertext c^*:

- Decryption request (2nd step). When $\mathcal{A}$ queries $(\texttt{ciphertext}, c)$ proceed as follows.

  1. Let
  $$(f, c_1, ..., c_K) = \mathsf{Extract}(c).$$
  2. If
  $$c^* \in \{c_1, ..., c_K\} \tag{7}$$
  then return $\perp$ to $\mathcal{A}$.

  Otherwise return her $\mathsf{Dec}(c)$.

As such, the vCCA_{SC} game is exactly the single challenge CCA2 game, with the second step decryption oracle being augmented with case 1 above (which, in essence, filters out *all* byproducts of the challenge ciphertext). Let us emphasize that vCCA_{SC} security is defined and investigated in [MN24] only under the (strong) assumption that S is *correct*. However, let us also highlight that *the correctness assumption plays no role in the above definition which remains meaningful in the general regime where approximate FHE are allowed.*

As a last comment, let us underline that, although its definition seems intrinsically single-hop (e.g., limited to one homomorphic evaluation over fresh ciphertexts), vCCA_{SC} security can, at least in principle, be extended to the multi-hop setting by allowing recursive calls to $\mathsf{Extract}$ [MN24, Remark 2, p. 28]. Still, as emphasized in [MN24] and later in Sect. 6, coming up with practically credible constructions for achieving vCCA security limited to the single-hop setting is already quite challenging yet sufficient to cover a wide range of FHE use-cases.

3.3 vCCA^D security: definitions and first properties

Contrary to the original vCCA security game introduced and studied in [MN24] (which, again, we refer to as vCCA_{SC} in this paper), the vCCA^D game is a multiple challenge one. Due to subtleties that will soon be clear, *we first define it in the private key setting*, starting from the CPA^D game in Sect. 3.1 *without the evaluation oracle* and assuming, as in vCCA_{SC}, the existence of the same extractor.

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

- Decryption request. When $\mathcal{A}$ queries $(\texttt{ciphertext}, c)$ proceed as follows:

  1. Let
  $$(f, c_1, ..., c_K) = \mathsf{Extract}(c).$$
  2. If
  $$f(\text{left}(c_1), ..., \text{left}(c_K)) \neq f(\text{right}(c_1), ..., \text{right}(c_K)) \tag{8}$$
  then return $\perp$ to $\mathcal{A}$.

  Otherwise return her $\mathsf{Dec}(c)$.

[3] Of course, the vCCA_{SC} game has no evaluation oracle as the adversary performs the homomorphic evaluations on its own in both the private and public key setting.

Where for any ciphertext $c \in \mathcal{C}$ we define

$$\text{left}(c) = \begin{cases} S[i].m_0 & \text{if } \exists i : S[i].c = c, \\ \perp & \text{otherwise,} \end{cases} \tag{9}$$

as well as,

$$\text{right}(c) = \begin{cases} S[i].m_1 & \text{if } \exists i : S[i].c = c, \\ \perp & \text{otherwise,} \end{cases} \tag{10}$$

and with the convention that $f(m_1, ..., m_K) = \perp$ when $\exists i : m_i = \perp$. Note that if the left and right evaluations both give $\perp$, condition (8) is not satisfied and $\mathsf{Dec}(c)$ is returned to $\mathcal{A}$.

Remark 1. It is clear that any ciphertext accepted by the vCCA^D game decryption oracle is also accepted by the LOR-CCA2 game one but not vice-versa. For example, ciphertext $\mathsf{Eval}(\text{sum}, c^*, \mathsf{Enc}(1))$ is accepted by the LOR-CCA2 decryption oracle (but rejected by the vCCA^D one) and trivially allows an adversary to win the CCA2 game. It follows that vCCA^D security is a strict relaxation of CCA2 security (which is well-known to exclude malleability[4]). †

Remark 2. In the private key setting, well-formed fresh ciphertexts (including challenge ones) can only be obtained by means of (encryption or challenge) oracle requests. Therefore, all such ciphertexts are registered in the game state S. It then follows that for any ciphertext of the form

$$c = \mathsf{Eval}(f, \mathsf{Enc}(m_1), ..., \mathsf{Enc}(m_K))$$

we have that $\text{left}(c) \neq \perp$ and $\text{right}(c) \neq \perp$. However, when the correctness assumption does not hold, we may have that $\mathsf{Dec}(c) \neq f(\text{left}(c_1), ..., \text{left}(c_K))$ as well as $\mathsf{Dec}(c) \neq f(\text{right}(c_1), ..., \text{right}(c_K))$. †

If we compare the vCCA$_{\text{SC}}$ game in previous Sect. 3.2 and the single challenge variant, vCCA$^D_{\text{SC}}$, of the above game, vCCA$_{\text{SC}}$'s second step oracle filters out *all* byproducts of the challenge ciphertext whereas (single challenge) vCCA$^D_{\text{SC}}$ filters out only those byproducts which allow to disciminate which of the two challenge plaintexts was encrypted.

Remark 3. From the definition of the two games, it is clear that all the ciphertexts accepted by the vCCA$_{\text{SC}}$ decryption oracle are also accepted by the vCCA$^D_{\text{SC}}$ one (but not vice-versa). There are indeed two types of ciphertexts which are rejected by the vCCA$_{\text{SC}}$ decryption oracle but accepted by the vCCA$^D_{\text{SC}}$ one:

1. Ciphertexts obtained through a legit call to Eval over well-formed fresh ciphertexts (with one of them being the challenge ciphertext) for which condition (8) does not hold, e.g. $\mathsf{Eval}(f, c^*, \mathsf{Enc}(m_2), ..., \mathsf{Enc}(m_K))$ with

   $$f(m_0^*, m_2, ..., m_k) = f(m_1^*, m_2, ..., m_k),$$

   where m_0^* and m_1^* denote the two challenge plaintexts. For example, ciphertext $\mathsf{Eval}(\text{mul}, c^*, \mathsf{Enc}(0))$ fall into this category.

2. Ciphertexts obtained through a legit call to Eval over arbitrary ciphertexts, with one of them being the challenge ciphertext and at least one of the others being ill-formed.

†

Throughout this paper and in particular in the separation results we establish, we (almost) always exploit the first of the above two gaps. As will be seen in Sect. 6, the second gap will be closed in the constructions themselves by including the machinery necessary for the decryption *function* of the proposed schemes to reply $\perp$ when given either an ill-formed ciphertext or an evaluated ciphertext over non well-formed ones.

[4] Additionally, it is well known that LOR-CCA2 security is equivalent to (single challenge) CCA2 (a.k.a., FTG-CCA2) in both the public key [BDPR98] and private key [BDJR97] settings.

3.3.1 Defining vCCA^D security in the public key case.

In the public key setting, the adversary can generate well-formed fresh ciphertexts independent of the challenge on his or her own. So only challenge-dependent fresh ciphertexts are guaranteed to be registered in the game state. In order to perform the left and right cleartext evaluations we therefore need a mean to access the messages that were given as inputs to the encryption function for well-formed ciphertexts that the adversary generated by his or herself. Note that, when the correctness assumption does not hold (which is also the regime under which we are willing to operate in this paper), these inputs cannot be recovered by merely decrypting those ciphertexts within the vCCA^D game decryption oracle.

Therefore, to define the vCCA^D game in the public key setting we need an additional extractor, denoted $\mathsf{Extract}'$, for recovering the encryption function inputs for fresh well-formed ciphertexts i.e.,

$$\mathsf{Extract}'(c) = \begin{cases} (m;r) & \text{when } c := \mathsf{Enc}(m,r), \\ \bot & \text{otherwise.} \end{cases} \tag{11}$$

Following this, *in the public key setting*, the vCCA^D game decryption oracle is then defined similarly to the private key case but with the notable difference that the left and right functions are replaced by the left$'$ and right$'$ functions defined as

$$\text{left}'(c) = \begin{cases} S[i].m_0 & \text{if } \exists i : S[i].c = c, \\ \mathsf{Extract}'(c).m & \text{otherwise,} \end{cases}$$

as well as,

$$\text{right}'(c) = \begin{cases} S[i].m_1 & \text{if } \exists i : S[i].c = c, \\ \mathsf{Extract}'(c).m & \text{otherwise.} \end{cases}$$

Following this, we however emphasize that, as we shall later see in Sect. 6, among the two public key vCCA$_{\text{SC}}$ scheme construction blueprints considered in [MN24], only the one in which fresh ciphertexts are defined as the association of an FHE ciphertext encrypting m by means of randomness r and another ciphertext encrypting the concatenation of m and r under a CCA2-secure encryption scheme is applicable in the general case where the correctness assumption may not hold. In this approach the well-formedness of fresh ciphertext is verified by first decrypting the companion CCA2 ciphertext to recover m and r and then checking that the associated FHE ciphertext is indeed equal to $\mathcal{E}_H.\mathsf{Enc}(m,r)$. This, in essence, provides the additional extractor, $\mathsf{Extract}'$, expected in the above definition.

3.3.2 vCCA^D security vs CPA^D security.

As a warm-up, let us now prove a first separation result between CPA^D and vCCA^D security.

Lemma 1. *vCCA^D security implies CPA^D security.*

Proposition 1 (CPA^D $\not\Longrightarrow$vCCA^D)**.** *If there exists a vCCA^D-secure scheme S, then there exists a scheme S' which is CPA^D-secure but not vCCA^D-secure.*

Proof. Let us start from a vCCA^D-secure scheme $S = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$. We now consider the scheme S' which is exactly S except that we modify the decryption function such that it leaks sk for a given randomly chosen input $c^\triangle \in \mathcal{C}$ i.e.,

$$\mathsf{Dec}'(c) = \begin{cases} \mathsf{sk} & \text{if } c = c^\triangle, \\ \mathsf{Dec}(c) & \text{otherwise.} \end{cases}$$

Additionally, S' public material now includes $c^{\triangle}$. The CPA^D security of S' then follows from the CPA^D security of S (by Lemma 1) as well as the fact that the CPA^D game decryption and evaluation oracles take state indices rather than ciphertexts as argument. As a consequence, a CPA^D adversary against S' cannot add $c^{\triangle}$, nor any byproduct of $c^{\triangle}$ obtained by means of homomorphic operations, to the game state (with non-negligible probability). Yet, S' is trivially vCCA^D-insecure, as a vCCA^D adversary against S' can submit $c^{\triangle}$ to the vCCA^{D}game decryption oracle and get sk in return. □

This simple proof pattern will occur several times in this paper.

4 Relations among the single challenge notions

We consider in this section the single challenge variants of vCCA^D and vCCA which we respectively denote vCCA$^D_{\mathrm{SC}}$ and vCCA$_{\mathrm{SC}}$. Following Sect. 3.2, in these single challenge variants, we allow only one challenge request with $m_0 \neq m_1$. The challenge ciphertext and the associated messages are respectively denoted c^*, m_0^* and m_1^*. We further consider in this section the single challenge variant of CPA^D which we denote CPA$^D_{\mathrm{SC}}$. In the single challenge case, we can further meaningfully split CPA$^D_{\mathrm{SC}}$ in CPA$^D_1$ and CPA$^D_2 \equiv$ CPA$^D_{\mathrm{SC}}$. Analogously to the distinction between CCA1 and CCA2, in CPA$^D_1$, the CPA$^D_{\mathrm{SC}}$ decryption oracle closes after the challenge request is performed.

4.1 Relations between single-challenge variants of CPA^D

Because CPA^D security collapses onto CPA security in the correct regime, then CPA$^D_1$ is equivalent to CPA$^D_2$ in that regime. In the general regime, however, we have the following separation result (considering that CPA$^D_2$ trivially implies CPA$^D_1$).

Proposition 2 (CPA$^D_1 \not\Rightarrow$CPA$^D_2$)**.** *If there exists an FHE scheme S which is CPA$^D_2$-secure, then there exists an FHE encryption scheme S' which is CPA$^D_1$-secure but CPA$^D_2$-insecure.*

Proof. Let us consider a CPA$^D_2$-secure FHE scheme $S = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$. Consider the approximate scheme $S' = (\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Eval}')$ built from S such that

$$\mathsf{Enc}'(m) = \mathsf{Enc}(m + g(m)),$$

where g is some function such that $g(0) \neq 0$, and KeyGen', Dec' and Eval' are similar to those of S.

S' is CPA$^D_1$-secure. Indeed, let us show that a CPA$^D_1$ adversary $\mathcal{A}$ against S' allows to build a CPA$^D_2$ adversary $\mathcal{B}$ against S. This reduction is simple: as $\mathcal{B}$ can simulate a CPA$^D_1$ game encryption request on m by sending $m + g(m)$ to the CPA$^D_2$ challenger encryption oracle, and can simulate the challenge request with $m_0 \neq m_1$ by sending $m_0 + g(m_0)$ and $m_1 + g(m_1)$ to the CPA$^D_2$ challenger. Both evaluation and decryption requests from $\mathcal{A}$ are transferred as is by $\mathcal{B}$. As a CPA$^D_1$ adversary, $\mathcal{A}$ does not perform any decryption request after its unique challenge request (and $\mathcal{B}$ then just replies $\perp$ to such requests).

S' is not CPA$^D_2$-secure. The adversary issues the *unique* encryption request with $m_0 \neq m_1$ to get

$$m_0^*, m_1^*, \mathsf{Enc}'(m_b^*),$$

say with $m_0^* = 0$ and $m_1^* = K$. It subsequently asks for an encryption of 0, c_0, and then asks for the computation of

$$c_{\mathrm{mul}} = \mathsf{Eval}'(\mathrm{mul}, c_0, c^*)$$

which the CPA_2^D decryption oracle accepts as it is associated to the triplet $(0, 0, c_{\mathrm{mul}})$ in the game state (c_{mul} is an encryption of 0 with respect to S'). Now, recall the definition of ε and p_ε in Eq. 2 (p. 6). Then, assuming that the adversary has chosen K such that

$$\|g(0)^2 - (K + g(K))g(0)\|_\infty > 2\varepsilon, \tag{12}$$

it can decide that $b = 0$ when

$$\|\mathsf{Dec}'(c_{\mathrm{mul}}) - g(0)^2\|_\infty \leq \varepsilon$$

and $b = 1$ otherwise, and win the CPA_2^D game with with probability at least p_ε. □

As an informal illustrative example for the above proof (assuming for simplicity sake $\mathcal{P} = \mathbb{Z}_t$, for some plaintext modulus $t >> \varepsilon$), we can choose $g(m) = 1$ for all m. So, when $b = 0$, $\mathsf{Dec}(c_{\mathrm{mul}})$ falls in $I_0 = [1-\varepsilon, 1+\varepsilon]$ and when $b = 1$ it falls in $I_K = [K+1-\varepsilon, K+1+\varepsilon]$. Then, if we choose $m_1^* = K > 2\varepsilon$, we get $I_0 \cap I_K = \varnothing$, ensuring condition (12) above.

Note that this result is different from Proposition 2 in [LM21] which establishes that there exists (approximate) FHE schemes which are non-adaptive CPA^D secure while being adaptive CPA^D insecure. Indeed, as already pointed in Sect. 3.1, there is a slight difference between the notion of adaptability as understood in the multiple-challenge context of [LM21] (the adversary performs all its requests at once) and that which is usually assumed between single-challenge CCA1 and CCA2 (the adversary performs all its decryption requests before the challenge is published).

4.2 Relations between $\mathrm{vCCA}_{\mathrm{SC}}$ and single-challenge variants of CPA^D

Let us emphasize that the issue of approximate schemes is only succinctly and informally discussed in [MN24]. Indeed, from a construction point of view, that paper claims to define blueprints for constructing vCCA-secure schemes from "state-of-the-art FHE such as TFHE or CKKS with the caveat that approximate FHE schemes need to be CPA^D-secure"[5]. It turns out that the results in this section clarify the relationship between $\mathrm{vCCA}_{\mathrm{SC}}$ security and CPA^D security in the general regime where approximate FHE are allowed: $\mathrm{vCCA}_{\mathrm{SC}}$ in fact requires much less than full-blown CPA^D security but rather its weaker "CCA1 style" variant, CPA_1^D. Implicitly, we consider here a generalization of $\mathrm{vCCA}_{\mathrm{SC}}$ security beyond the correctness assumption. However, as argued in Sect. 3.2, in terms of security game definition, there is no dependency on the correctness assumption. Indeed, in [MN24], that assumption only steps in for proving the $\mathrm{vCCA}_{\mathrm{SC}}$ security of the proposed constructions.

Proposition 3. *$\mathrm{vCCA}_{\mathrm{SC}}$ security implies CPA_1^D security.*

Proof. By definition, a CPA_1^D adversary can only perform decryption requests which are independent of the challenge ciphertext. It thus follows that any request performed by a CPA_1^D adversary can also be performed by a $\mathrm{vCCA}_{\mathrm{SC}}$ one. □

Proposition 4 ($\mathrm{vCCA}_{\mathrm{SC}} \not\Longrightarrow \mathrm{CPA}_2^D$)**.** *If there exists a $\mathrm{vCCA}_{\mathrm{SC}}$-secure scheme S, then there exists a scheme S' which is $\mathrm{vCCA}_{\mathrm{SC}}$-secure and CPA_2^D-insecure.*

The proof is similar to the proof of Proposition 2 and deferred to Appendix B.1

Proposition 5 ($\mathrm{CPA}_2^D \not\Longrightarrow \mathrm{vCCA}_{\mathrm{SC}}$)**.** *If there exists a CPA_2^D-secure scheme S, then there exists a scheme S' which is CPA_2^D-secure and $\mathrm{vCCA}_{\mathrm{SC}}$-insecure.*

The proof is essentially identical to that of Proposition 1 and deferred to Appendix B.2.

[5] Following the recent attacks in [CSBB24, CCP+24], the authors of [MN24] further updated their ePrint version to put additional emphasis on the FHE correctness assumption (see Remark 1 on p. 5 and Sect. 5.4).

4.3 Relations between $\mathrm{vCCA}_{\mathrm{SC}}$ and $\mathrm{vCCA}^{D}_{\mathrm{SC}}$ security

In this section, we establish the relationships between $\mathrm{vCCA}_{\mathrm{SC}}$ security (recall that only $\mathrm{vCCA}_{\mathrm{SC}}$ is studied in [MN24] and, as such, only denoted vCCA) and $\mathrm{vCCA}^{D}_{\mathrm{SC}}$ in both the correct regime and the general regime where approximate FHE are allowed. In a nutshell, we establish that, although the two notions are equivalent in the correct regime, $\mathrm{vCCA}^{D}_{\mathrm{SC}}$ security is strictly stronger than $\mathrm{vCCA}_{\mathrm{SC}}$ is the general case.

Lemma 2. *$vCCA^{D}_{SC}$ security implies $vCCA_{\mathrm{SC}}$ security.*

Proof. By definition of the two games, all decryption requests accepted by the $\mathrm{vCCA}_{\mathrm{SC}}$ decryption oracle are also accepted by the $\mathrm{vCCA}^{D}_{\mathrm{SC}}$ one (recall also Remark 3 on p. 10). It thus follows that any request performed by a $\mathrm{vCCA}_{\mathrm{SC}}$ adversary can also be performed by a $\mathrm{vCCA}^{D}_{\mathrm{SC}}$ one. □

Note that the above implication holds in the general regime i.e., independently of the correctness assumption.

We then prove a first result showing that $\mathrm{vCCA}_{\mathrm{SC}} \not\Longrightarrow \mathrm{vCCA}^{D}_{\mathrm{SC}}$ in the correct regime, under some condition on the probability that an adversary may bypass plaintext awareness.

Proposition 6. *Let S be a $vCCA_{\mathrm{SC}}$-secure scheme and let $\mu^{\perp}$ denotes the probability that $\mathsf{Dec}(u) = \perp$ for $u \xleftarrow{\$} \mathcal{C}$, then, under the correctness assumption, there exists an adversary against the $vCCA^{D}_{SC}$-security of S which achieves advantage $(1-\mu^{\perp})(1-1/|\mathcal{P}|)$.*

Proof. Consider a $\mathrm{vCCA}_{\mathrm{SC}}$-secure scheme $S = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ and assume $m^{*}_{0} = 0$, $m^{*}_{1} = 1$ and $c^{*} = \mathsf{Enc}(m^{*}_{b})$. We now consider the following $\mathrm{vCCA}^{D}_{\mathrm{SC}}$ attack against S. First, $\mathcal{A}$ picks a ciphertext c_{rnd} uniformly at random in $\mathcal{C}$. It then performs,

$$c_{\mathrm{mul}} = \mathsf{Eval}(\mathrm{mul}, c^{*}, c_{\mathrm{rnd}}).$$

where c_{mul} (as well as c_{rnd}) decrypts to $\perp$ with probability $\mu^{\perp}$[6]. Now, since it is a byproduct of the challenge ciphertext via an invocation of Eval, ciphertext c_{mul} is rejected by the $\mathrm{vCCA}_{\mathrm{SC}}$ game decryption oracle. However, c_{mul} is accepted by the $\mathrm{vCCA}^{D}_{\mathrm{SC}}$ game decryption oracle: since c_{rnd} cannot be part of the $\mathrm{vCCA}^{D}_{\mathrm{SC}}$ game state (as $\mathcal{A}$ does not know the associated plaintext), then neither can be c_{mul}. When $\mathsf{Dec}(c_{\mathrm{mul}}) = 0$, then $\mathcal{A}$ decides that $b = 0$ and, under the correctness assumption, wins the $\mathrm{vCCA}^{D}_{\mathrm{SC}}$ game with probability $1 - \frac{1}{|\mathcal{P}|}$. Hence the claim. □

Note that thanks to condition (2), this proposition can also easily be generalized to the general regime which also include approximate FHE schemes (in this case $\mathcal{A}$ decides that $b = 0$ when $\|\mathsf{Dec}(c_{\mathrm{mul}})\|_{\infty} \leq \varepsilon$ and wins the game with advantage $p_{\varepsilon}(1-\mu^{\perp})(1-1/|\mathcal{P}|)$). However this result is of limited interest as the above attack leads to a non-negligible advantage only if S is such that $1-\mu^{\perp} > \mathrm{neg}(\lambda)$[7]. However, as already discussed in Sect. 3.2 (recall Remark 3, page 10 and its follow up discussion) all the $\mathrm{vCCA}_{\mathrm{SC}}$-secure constructions proposed in [MN24] and revisited in Sect. 6 include the machinery necessary for their decryption *function* to return $\perp$ when given either an ill-formed ciphertext (which will then be the case of c_{rnd} with at least $1 - \mathrm{neg}(\lambda)$ probability) or an evaluated ciphertext over non well-formed ones (which is then the case of c_{mul}). As a consequence, it is interesting to study the relationship between $\mathrm{vCCA}_{\mathrm{SC}}$-security and $\mathrm{vCCA}^{D}_{\mathrm{SC}}$-security *under the well-formedness assumption* whereby the adversary is limited to exploit only legit ciphertexts (well-formed fresh ciphertexts or ciphertexts derived from fresh well-formed ciphertexts via legit homomorphic evaluations).

[6] Following the consistency assumption (4) (Sect. 2).

[7] An example of such a scheme could be that of a LWE scheme where $\perp$ is returned when $|e'| \geq T$ for some threshold value T, with $e' = b - \left(\langle a, \mathsf{sk}\rangle + \Delta \left\lceil \frac{((b - \langle a, \mathsf{sk}\rangle))}{\Delta} \right\rfloor\right)$. With $q \leq O(\lambda)$, we then indeed get $(1-\mu^{\perp}) = \frac{4T}{q} > \mathrm{neg}(\lambda)$.

Proposition 7. *Under both the FHE correctness and (the above) well-formedness assumptions, vCCA$_{\mathrm{SC}}$ security is equivalent to vCCA$^{D}_{SC}$ security.*

Proof. Following Lemma 2 we have that vCCA$^{D}_{\mathrm{SC}}$ security implies vCCA$_{\mathrm{SC}}$ security. For the other direction, we will now show that given a successful adversary $\mathcal{A}$, working under the well-formedness assumption, against the vCCA$^{D}_{\mathrm{SC}}$ security of a scheme S, there exists a successful adversary $\mathcal{B}$, also working under the well-formedness assumption, against the vCCA$_{\mathrm{SC}}$ security of S which uses $\mathcal{A}$ as a subroutine. Then $\mathcal{A}$ can issue the following requests which $\mathcal{B}$ emulates as follows:

- When $\mathcal{A}$ issues a vCCA$^{D}_{\mathrm{SC}}$ game encryption request for plaintext m, then $\mathcal{B}$ issues a vCCA$_{\mathrm{SC}}$ game encryption request to get ciphertext c which it returns to $\mathcal{A}$.
- When $\mathcal{A}$ issues the *unique* vCCA$^{D}_{\mathrm{SC}}$ game challenge request for plaintexts m_0^*, m_1^* ($m_0^* \neq m_1^*$), then $\mathcal{B}$ issues a vCCA$_{\mathrm{SC}}$ game challenge request to get ciphertext c^* which it returns to $\mathcal{A}$. Additionally, $\mathcal{B}$ stores m_0^*, m_1^* as well as c^*.
- When $\mathcal{A}$ issues a vCCA$^{D}_{\mathrm{SC}}$ game decryption request for *well-formed* (recall that $\mathcal{A}$ works under the well-formedness assumption) ciphertext c, then $\mathcal{B}$ runs the extractor over c to get
$$(f, c_1, ..., c_K) = \mathsf{Extract}(c),$$
and then proceeds as follows:
  - If $c^* \notin \{c_1, ..., c_K\}$, $\mathcal{B}$ simply issues a vCCA$_{\mathrm{SC}}$ game decryption request and forward the resulting plaintext (which is necessarily not $\perp$) to $\mathcal{A}$.
  - Otherwise (assuming wlog that the challenge ciphertext appears only once as the first position argument), $\mathcal{B}$ issues vCCA$_{\mathrm{SC}}$ game decryption requests for $c_2, ..., c_K$, getting plaintexts $m_2, ..., m_K$ which were the associated encryption function inputs (because of the correctness assumption). Then $\mathcal{B}$ compute $r_0 = f(m_0^*, m_2, ..., m_K)$ and $r_1 = f(m_1^*, m_2, ..., m_K)$ and returns r_0 when $r_0 = r_1$ (in this case, under the correctness assumption, $r_0 = r_1 = \mathsf{Dec}(c)$) and $\perp$ otherwise.

Thus, $\mathcal{B}$ can duly simulate all the vCCA$^{D}_{\mathrm{SC}}$ game requests issued by $\mathcal{A}$. □

Note that this proof is performed under the well-formedness assumption whereby all ciphertexts exploited by the adversary are well-formed. If we take a glimpse at the constructions that will be discussed in Sect. 6, this means that the SNARK proofs of correct homomorphic evaluations are always valid and that the associated extractor always exists. Thus, for the Designated-Verifier case, $\mathcal{B}$ does not need access to a verification oracle. Also, because this proof is done under the correctness assumption, $\mathsf{Extract}'$ (as defined in Sect. 3.2, Eq. (11)) is not needed for $\mathcal{B}$ to recover the encryption function inputs (as these can be recovered via calls to the decryption oracle of the vCCA$_{\mathrm{SC}}$ game that $\mathcal{B}$ is playing against).

In the general case, however, it turns out that the two notions can be separated as established by the following proposition.

Proposition 8 (vCCA$_{\mathrm{SC}} \not\Longrightarrow$ vCCA$^{D}_{\mathrm{SC}}$)**.** *In the general regime, if there exists a vCCA$^{D}_{SC}$-secure scheme S, then there exists a scheme S' which is vCCA$_{\mathrm{SC}}$-secure but vCCA$^{D}_{SC}$-insecure, even against an adversary working under the well-formedness assumption.*

The proof works similarly to that of Proposition 2 (and Proposition 4) and is deferred to Appendix B.3.

Following Lemma 2 and Proposition 8 we can conclude that vCCA$^{D}_{\mathrm{SC}}$ security is strictly stronger than vCCA$_{\mathrm{SC}}$ security in the general regime.

5 Relations among the multiple challenge notions

5.1 Relations between $\mathbf{CPA}^D_{\mathbf{SC}}$ and $\mathbf{CPA}^D$ security

In this section, we first focus on CPA^D and study the relationship between the single and multiple-challenge variants of this notion. This is an interesting question as, unless the FHE scheme is restricted to the evaluation of univariate functions, the usual hybrid argument e.g. in [BDJR97] (theorem 4) for showing the equivalence (up to an increase in advantage linear in the number of challenge ciphertexts) between FTG-CPA[8] (resp. FTG-CCA) and LOR-CPA (resp. LOR-CCA) does not work directly. This is so because an adversary confronted to a hybrid game (where the encryption oracle replies according to $b = 0$ up to a random point after which it replies according to $b = 1$) can detect the transition between the first and second phase since ciphertexts from the two phases may interact via evaluation requests. The relationship between single and multiple-challenge variants of CPA^D was also explicitly left as an open question in [LM21].

Then, we also establish that, for CPA^D, the single and multiple challenge variants are (without surprise) equivalent in the correct regime and, more interestingly, that the two notions can be separated in the general regime (which is the non-trivial CPA^D setting).

We first recall the following well-known theorem from [BDJR97].

Theorem 1. *For any encryption scheme $S = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, LOR-CPA is equivalent to FTG-CPA*[9]*.*

For CPA^D security, in the correct FHE regime, we have the following equivalence. This equivalence is not surprising since CPA^D security collapses onto CPA security for correct FHE.

Proposition 9. *For any correct FHE scheme, $S = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$, CPA^D security is equivalent to CPA^D_{SC} security.*

Proof. For a correct FHE scheme, it is well-known that LOR-CPA^D is equivalent to LOR-CPA [LM21]. This means that an adversary to the LOR-CPA^D game has exactly the same advantage as an adversary to *the LOR-CPA^D game without the decryption oracle*, which is the same as the LOR-CPA game plus the evaluation oracle (recall that the LOR-CPA^D game decryption oracle, Sect. 3.1, takes indices from the game state rather than ciphertexts as input). Let us call this the LOR-CPA^E game ("CPA with an evaluation oracle"). It is easy to see that LOR-CPA^E is exactly LOR-CPA. For the same reasons, $\mathrm{CPA}^D_{\mathrm{SC}}$ is equivalent to FTG-CPA. The claim then follows from theorem 1 above. □

In the general regime where approximate FHE are allowed, things are more interesting as we can actually separate the two notions. We start by proving the separation in the special case of additive FHE scheme.

Proposition 10. *In the general regime, if there exists an additive HE scheme S which is CPA^D-secure, then there exists an additive HE scheme S' which is CPA^D_{SC}-secure and CPA^D-insecure.*

Proof. So let us start with a CPA^D-secure *additive* HE scheme

$$S = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}).$$

[8] Recall that Find-Then-Guess (FTG) is the terminology for single challenge security notions in the foundational papers [BDJR97, BDPR98].

[9] More precisely [BDJR97] established that the advantage of a LOR-CPA (resp. LOR-CCA) adversary is bounded by $q_e \alpha_{\mathrm{sc}}$, where q_e is an upper bound the number of encryption queries with $m_0 \neq m_1$ and α_{sc} is the advantage of an FTG-CPA (resp. FTG-CCA) adversary.

Then, consider the scheme $S' = (\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Eval}')$ such that

$$\mathsf{Enc}'(m) = \mathsf{Enc}(m + g(m)),$$

where g is some non linear function such that $g(a+b) \neq g(a)+g(b)$[10]. Additionally KeyGen', Dec' and Eval' are the same as those of S. The proof is further done under the mild assumption that the plaintext space of S admits no divisor of 0.

S' is CPA^D_{SC}-secure. First of all, let us remark that the CPA^D_1 security of S' follows from the CPA^D security of S and the fact that the approximation noise $g(m)$ is independent of S secret key material (which is the only information not available to a CPA^D_1 adversary). So to break the $\mathrm{CPA}^D_{\mathrm{SC}}$ security of S', an attacker has to exploit the challenge ciphertext. However, after the adversary issues his or her unique challenge request with $m_0 \neq m_1$ to get m_0^*, m_1^* and $c^* = \mathsf{Enc}'(m_b^*)$, it can only

- asks for a decryption of c^*, which is rejected by the $\mathrm{CPA}^D_{\mathrm{SC}}$ decryption oracle since $m_0^* \neq m_1^*$,
- asks for a decryption of $c_{\mathrm{sum}} = \mathsf{Eval}'(\mathrm{sum}, c^*, c^{(1)}, ..., c^{(K)})$ where[11] all the $c^{(i)}$'s are such that $m_0^{(i)} = m_1^{(i)}$. The decryption of c_{sum} is also blocked by the $\mathrm{CPA}^D_{\mathrm{SC}}$ decryption oracle since
$$m_0^* + \sum_i m_0^{(i)} \neq m_1^* + \sum_i m_1^{(i)}.$$

So the adversary can learn nothing on b and the $\mathrm{CPA}^D_{\mathrm{SC}}$-security of S' follows.

S' is not CPA^D-secure. Now let the adversary issue two challenge requests with $m_0 \neq m_1$ getting,

$$m_0^*, m_1^*, c^* = \mathsf{Enc}'(m_b^*),$$

as well as

$$m_0^\dagger, m_1^\dagger, c^\dagger = \mathsf{Enc}'(m_b^\dagger),$$

such that $Z = m_0^* + m_0^\dagger = m_1^* + m_1^\dagger$ and $g(m_0^*) + g(m_0^\dagger) \neq g(m_1^*) + g(m_1^\dagger)$. The adversary then computes

$$c_{\mathrm{sum}} = \mathsf{Eval}'(\mathrm{sum}, c^*, c^\dagger)$$

which the CPA^D decryption oracle accepts since the left and right evaluations both give Z i.e., c_{sum} is an encryption of Z with respect to S'. However, with respect to S, c_{sum} is an encryption of $Z + g(m_b^*) + g(m_b^\dagger)$, hence the adversary gets $\mathsf{Dec}'(c_{\mathrm{sum}}) = \mathsf{Dec}(c_{\mathrm{sum}})$ such that

$$\|\mathsf{Dec}'(c_{\mathrm{sum}}) - Z - g(m_b^*) - g(m_b^\dagger)\|_\infty \leq \varepsilon$$

as a result of a decryption request on c_{sum} (recall the definition of ε and p_ε in Eq. 2, p. 6). Since $g(m_b^*) + g(m_b^\dagger) \neq g(m_b^* + m_b^\dagger)$ and $g(m_0^*) + g(m_0^\dagger) \neq g(m_1^*) + g(m_1^\dagger)$ (and further assuming that $\|G_0 - G_1\|_\infty > 2\varepsilon$ with $G_b = g(m_b^*) + g(m_b^\dagger)$), the adversary recovers b with probability at least p_ε, leading the claim. □

To informally illustrate this (considering for simplicity sake $\mathcal{P} = \mathbb{Z}_t$, for some large *prime* plaintext modulus t), assuming $\varepsilon << 10000$ for the sake of an example, we can consider the following concrete setup, where $B = 10000$ and $g(x) = 2\varepsilon\lfloor x/B \rceil^2$ [12]. Then, as

[10] More precisely, we need g such that there exists a, b, a' and b' such that $a + b = a' + b'$ while $g(a) + g(b) \neq g(a') + g(b')$.

[11] Under the assumption that $\forall m_0^*, m_1^* \in \mathcal{P}$ such that $m_0^* \neq m_1^*$ and $\forall k \in \mathbb{N}^*$ we have $k * m_0^* \neq k * m_1^*$, we can assume, without loss of generality that c^* appears only once and in the first place in the $\mathsf{Eval}'(\mathrm{sum}, ...)$ arguments. Note that this requirement can easily be satisfied by $\mathcal{P}$ being a field.

[12] Stricto sensu, to ensure the boundedness of $g(x)$, it would be preferable to choose $g(x) = 2\varepsilon \min\left(L, \lfloor x/B \rceil^2\right)$ with e.g. $L = 5$ for our example to work. That way, S' is also an approximate FHE scheme consistent with our definition (2) (p. 6).

above, the adversary chooses $m_0^\star = m_0^\dagger = 10000$ as well as $m_1^\star = 0$ and $m_1^\dagger = 20000$. With these parameters, $Z = 20000$ and $\mathsf{Dec}'(c_{\text{sum}})$ belongs (with probability p_ε) to $[20000 + 3\varepsilon, 20000 + 5\varepsilon]$ when $b = 0$, or to $[20000 + 7\varepsilon, 20000 + 9\varepsilon]$ when $b = 1$.

Because the proof of Proposition 10 relies strongly on the additive-only property of the scheme to prove that single challenge attacks are not possible, the fact of being FHE (i.e. allowing also multiplication) could prevent the separation. Hence, we now further establish the separation result without the restriction to additive HE schemes.

Proposition 11 ($\text{CPA}^D_{\text{SC}} \not\Longrightarrow \text{CPA}^D$)**.** *In the general regime, if there exists an FHE scheme S which is CPA^D-secure, then there exists an FHE scheme S' which is CPA$^D_{SC}$-secure and CPA^D-insecure.*

Proof. So let us start with a CPA^D-secure FHE scheme

$$S = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, \mathsf{Mul}).$$

Then, consider the scheme $S' = (\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Add}', \mathsf{Mul}')$ such that

$$\mathsf{Enc}'(m) = (\mathsf{Enc}(m), \mathsf{Enc}(g(m))) = (c_0, c_1) = c,$$

(with g being the same as in the proof of the previous Proposition 10) and

$$\mathsf{Dec}'(c) = \mathsf{Dec}(c_0) + \mathsf{Dec}(c_1).$$

Now we consider the following homomorphic addition and multiplication operators:

$$\mathsf{Add}'(c, c') = (\mathsf{Add}(c_0, c_0'), \mathsf{Add}(c_1, c_1'))$$

and

$$\mathsf{Mul}'(c, c') = (\mathsf{Mul}(c_0, c_0'), \mathsf{Enc}(0)).$$

Essentially, in S', the approximation noise is encrypted separately to the message (to make both easily separable) and the multiplication operator resets that noise. So as soon as the adversary performs a multiplication, it closes the information channel that function g opens. Although S' is fully homomorphic, we therefore end up in the same conditions as in the proof of Proposition 10. Indeed, in any successful CPA^D_{SC} or CPA^D attack involving both homomorphic additions and multiplications, the multiplications are redundant. □

Since CPA^D security trivially implies CPA^D_{SC} security, the following result is a direct consequence of Proposition 11.

Corollary 1. *In the general regime, CPA^D is strictly stronger than CPA$^D_{SC}$.*

Note that the above proposition (partly) settles the question of the relationship between single and multiple-challenge CPA^D security that was left open in [LM21] (p. 14): "It remains an interesting open question to find out the relationship between $(q;\ell)$-IND-CPA^D and $(q;\ell+1)$-IND-CPA^D securities (and same for SIM-CPA^D)." (in their notations, ℓ is the number of queries to the encryption oracle with $m_0 \neq m_1$ and q the number of decryption queries). Indeed, we have shown above that, even for $q = 1$, there exists homomorphic schemes which are $(q;1)$-CPA^D-secure while being $(q,2)$-CPA^D-insecure (in the notations of [LM21]). Of course, to completely settle the above question we also need to prove separation or equivalence of (q,ℓ)-CPA^D-security and $(q,\ell+1)$-CPA^D-security with $\ell > 2$. We solve this remaining question in Appendix D, where we establish that these notions can be separated (Proposition 19).

5.2 Relations between vCCA_{SC} and vCCA security

Recall that [MN24] defines and studies only vCCA_{SC} security (also recall that in that paper it is simply referred to as vCCA). The question of the relationship between vCCA_{SC} security and LOR-vCCA security (or simply vCCA security with our present conventions) therefore deserves to be settled. So let LOR-vCCA denote the multiple challenges variant of vCCA in which the decryption oracle condition (7) is replaced by

$$C^* \cap \{c_1, ..., c_K\} \neq \emptyset, \tag{13}$$

where C^* is the set of challenge ciphertexts. We then have the proposition below.

Proposition 12. *vCCA security is equivalent to vCCA_{SC} security.*

Proof. The standard hybrid argument, e.g. in the proof of [BDJR97, Theorem 4] (which corresponds to Theorem 1 for both CPA and CCA), holds without modification, since an adversary confronted to a hybrid game (where the encryption oracle replies according to $b = 0$, up to a random point after which it replies according to $b = 1$) *cannot* detect the transition between the first and second phases. Indeed, although challenge ciphertexts from both phases may interact via homomorphic evaluations, condition (13) above guarantees that such interactions lead to ciphertexts rejected by the above LOR-vCCA decryption oracle. □

Note that in the above proof, we make no assumption about the correctness of the FHE scheme, so the above equivalence holds in the general regime.

As a consequence of Proposition 8, this establishes that the single challenge variant of vCCA^D security, $\text{vCCA}^D_{\text{SC}}$, is already strictly stronger than the multiple challenge variant of vCCA.

5.3 Relations between $\text{vCCA}^D_{\text{SC}}$ and vCCA^D security

Lastly, we now unveil the relationship between the single and multiple challenge variant of vCCA^D security. Similarly to the CPA^D case, there is a distinction between the correct and general regime.

Proposition 13. *In the correct regime, vCCA^D is equivalent to vCCA^D_{SC}.*

Proof. The claim follows directly from Proposition 7 (and its straightforward generalization to the multiple challenges variants of vCCA and vCCA^D) as well as Proposition 12. □

We now turn to the general regime and, as in Sect. 5.1, first consider the case of linearly homomorphic schemes.

Proposition 14. *In the general regime, if there exists an additive HE scheme S which is vCCA^D-secure, then there exists an additive HE scheme S' which is vCCA^D_{SC}-secure and vCCA^D-insecure.*

The proof proceeds similarly to that of Proposition 10 and is deferred to Appendix B.4.

Proposition 15 ($\text{vCCA}^D_{\text{SC}} \not\Longrightarrow \text{vCCA}^D$)**.** *In the general regime, if there exists an FHE scheme S which is vCCA^D-secure, then there exists an FHE scheme S' which is vCCA^D_{SC}-secure and vCCA^D-insecure.*

Proof. Identical to that of Proposition 11. □

Since vCCA^D security trivially implies $\text{vCCA}^D_{\text{SC}}$ security, the following result is a direct consequence of Proposition 15.

Corollary 2. *In the general regime, vCCA^D is strictly stronger than vCCA^D_{SC}.*

It follows that vCCA^D security is the strongest CCA security notion so far known to be achievable by FHE in the general regime.

6 Construction blueprints

In this section, we revisit three out of the four[13] construction blueprints proposed in [MN24] to leverage CPA-secure and correct FHE schemes into $\mathrm{vCCA}_{\mathrm{SC}}$-secure schemes, and study both their applicability in the general regime where approximate FHE are allowed as well as their vCCA^{D} security. Additionally, we propose a new construction blueprint for the public key setting, which we refer to as *Encrypt-then-Prove*. As such, we emphasize that all but the latter blueprints are not new.

6.1 Private key constructions

We first consider the Encrypt-then-Sign construction blueprint proposed in [MN24]. The construction is built over a public-key FHE scheme $\mathcal{E}_H$, a public-key signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, as well as a (publicly verifiable or designated-verifier) SNARK, $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$, for langage

$$L_1 = \left\{ c_e | \exists f \in \mathcal{F}_H, \exists (c_1, \pi_1), ..., (c_K, \pi_K) \in \mathcal{C}^K, \begin{array}{l} \Sigma.\mathsf{Verify}(\Sigma.\mathsf{pk}, c_i, \pi_i), \forall i \\ c_e = \mathcal{E}_H.\mathsf{Eval}(\mathcal{E}_H.\mathsf{pk}, f, c_1, ..., c_K) \end{array} \right\}, \quad (14)$$

to obtain an encryption scheme $\mathcal{E}^{\star}_H$ defined as follows:

- $\mathcal{E}^{\star}_H.\mathsf{KeyGen}$: run $\mathcal{E}_H.\mathsf{KeyGen}$, $\Sigma.\mathsf{KeyGen}$, $\Pi.\mathsf{Setup}$, let $\mathsf{ek} = (\mathcal{E}_H.\mathsf{pk}, \Sigma.\mathsf{sk})$ as well as $\mathsf{sk} = (\mathcal{E}_H.\mathsf{sk}, \Sigma.\mathsf{pk}, [\Pi.\mathsf{vk}])$.
- $\mathcal{E}^{\star}_H.\mathsf{Enc}$: given $m \in \mathcal{P}$ generate ciphertext
$$(c, \pi) = (\mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk}, m), \Sigma.\mathsf{Sign}(\Sigma.\mathsf{sk}, c)).$$
- $\mathcal{E}^{\star}_H.\mathsf{Eval}$: given ciphertexts $(c_1, \pi_1), ..., (c_K, \pi_K)$ such that $\Sigma.\mathsf{Verify}(c_i, \pi_i), \forall i$, generate ciphertext (c_e, π_e) such that
$$c_e = \mathcal{E}_H.\mathsf{Eval}(\mathcal{E}_H.\mathsf{pk}, f, c_1, ..., c_K),$$
and
$$\pi_e = \Pi.\mathsf{Prove}(c_e, (f, (c_1, \pi_1), ..., (c_K, \pi_K))).$$
- $\mathcal{E}^{\star}_H.\mathsf{Dec}$: given ciphertext (c, π) return $\mathcal{E}_H.\mathsf{Dec}(\mathcal{E}_H.\mathsf{sk}, c)$ when $\Sigma.\mathsf{Verify}(\Sigma.\mathsf{pk}, c, \pi) = \mathsf{True}$ or $\Pi.\mathsf{Verify}([\Pi.\mathsf{vk}], c, \pi) = \mathsf{True}$, and $\perp$ otherwise.

Intuitively, the essence of this construction is to rely on a trusted encryption oracle that generates only well-formed ciphertexts with respect to $\mathcal{E}_H$ and signs them such that they are recognizable. As such, this construction is not public key, due to the presence of $\Sigma.\mathsf{sk}$ in ek. Also, let us emphasize that this construction satisfies the compactness property which is implicitly assumed for FHE as the size of the output of Eval is independent of the size of f and, in particular, of its arity, thanks to the succinctness of the SNARK. For completeness, we also provide in Appendix C a variant of this constructions which may be more practical, however at the expense of input privacy and compactness.

The above Encrypt-then-Sign construction was proved in [MN24] to lead a $\mathrm{vCCA}_{\mathrm{SC}}$-secure scheme from a CPA-secure *correct* FHE scheme, a SUF-CMA-secure signature scheme and a simulation-extractable SNARK (which implies the existence of an extractor as defined in Sect. 3.2). Their proof technique consists in showing that a successful $\mathrm{vCCA}_{\mathrm{SC}}$ attack over scheme $\mathcal{E}^{\star}_H$ implies a successful CCA2 attack against the *private key* (non homorphic) encryption scheme obtained by associating $\mathcal{E}_H$ and Σ. It turns out that

[13] We did not revisit their Naor-Yung-based blueprint, as it is only applicable when the underlying FHE is *perfectly* correct.

the security of this construction goes beyond this setting as we now prove that the above Encrypt-then-Sign blueprint in fact offers vCCA^D security beyond the correct FHE regime, as long as we instantiate it from a CPA^D-secure rather than CPA-secure/correct FHE. To do so, we proceed in the next proof with two steps of game hopping, respectively relying on the SUF-CMA security of the signature scheme and the straightline-extractability of the SNARK, followed by a final reduction towards the CPA^D security of the underlying FHE scheme.

Proposition 16 (Encrypt-then-Sign)**.** *Let $\mathcal{A}$ be an adversary against the vCCA^D security of $\mathcal{E}^\star_H$, then, under the assumption that Σ is SUF-CMA secure and Π is straightline-extractable, there exists an adversary $\mathcal{B}$ against the CPA^D security of $\mathcal{E}_H$ which uses $\mathcal{A}$ as a subroutine.*

Proof. We thus start with the vCCA^D security game between an adversary $\mathcal{A}$ and the challenger. For all the following games G_i, let us call O_i the event "adversary $\mathcal{A}$ outputs 1 in the game G_i". We now consider a sequence of games, starting from G_0, which is the vCCA^D game.

First game hop. Let G_1 denote the same game as G_0 except that the challenger replaces the signature verification for a given fresh ciphertext by checking whether it has generated it, i.e. starting from an initially empty state $S^\Sigma = []$:

- When it outputs a fresh ciphertext (c, π) following an encryption or a challenge request, it updates its internal state $S^\Sigma = [S^\Sigma; (c, \pi)]$.
- When it has to verify the signature of a fresh ciphertext (c, π), following a decryption request, it replies True if and only if $(c, \pi) \in S^\Sigma$.

Clearly, the two above games proceed identically unless the following failure event occurs: "$\mathcal{A}$ succeeds in forging a signature (which will be accepted in G_0 and denied in G_1)", then breaking the SUF-CMA security of Σ. Hence,

$$|P(O_0) - P(O_1)| \leq \mathsf{Adv}^{\text{SUF-CMA}}_{\mathcal{A},\Sigma}.$$

Second game hop. We now consider game G_2, which is the same as G_1 except that the challenger replaces all verifications of the proofs under Π by invoking a verification oracle and then the straightline extractor to get the witnesses it checks by itself, i.e.

- When it has to verify a proof associated with an evaluated ciphertext (c, π), following a decryption request, it first invoke a verification oracle to determine whether or not the proof is correct. Then it runs the extractor to get the witnesses $f, (c_1, \pi_1), ..., (c_K, \pi_K)$ (following language L_1) and checks the statement by itself (with the (c_i, π_i)'s signatures being checked as in G_1).

Then, G_1 and G_2 proceed identically unless the following failure event occurs: "$\mathcal{A}$ succeeds in forging a valid proof such that the extracted witness is invalid", thus breaking the knowledge soundness of Π. Hence,

$$|P(O_1) - P(O_2)| \leq \mathsf{Adv}^{\text{Knowledge Soundness}}_{\mathcal{A},\Pi}.$$

Let us now show that an adversary $\mathcal{B}$ against the CPA^D security of $\mathcal{E}_H$ can be built using $\mathcal{A}$ against G_2 (or equivalently G_0) as a subroutine.

Final reduction. We are now in the last game, where we build an adversary $\mathcal{B}$ against the CPA^D security, from the adversary $\mathcal{A}$: $\mathcal{B}$ initially runs $\Pi.\mathsf{Setup}$ and $\Sigma.\mathsf{KeyGen}$ and communicates the associated public material to $\mathcal{A}$. Additionally, $\mathcal{B}$ mimics the CPA^D game state and initially starts with an empty state $S^\mathcal{B} = []$. Then, given a ciphertext c, we denote by $\mathsf{idx}(c)$, its index in the game state $S^\mathcal{B}$ (which is the same as the index at which the ciphertext is stored in the CPA^D game state S). Then $\mathcal{A}$ can issue the following requests, which $\mathcal{B}$ emulates as follows:

- When $\mathcal{A}$ issues a vCCA^D game encryption request for plaintext m, then $\mathcal{B}$ issues a CPA^D game encryption request to get ciphertext c. $\mathcal{B}$ then does $S^{\mathcal{B}} := [S^{\mathcal{B}}; (m, m, c)]$, generates $\pi = \Sigma.\mathsf{Sign}(\Sigma.\mathsf{sk}, c)$ and return (c, π) to $\mathcal{A}$.
- When $\mathcal{A}$ issues a vCCA^D game challenge request for plaintexts m_0, m_1 ($m_0 \neq m_1$), then $\mathcal{B}$ issues a CPA^D game challenge request to get ciphertext c. $\mathcal{B}$ then does $S^{\mathcal{B}} := [S^{\mathcal{B}}; (m_0, m_1, c)]$, generates $\pi = \Sigma.\mathsf{Sign}(\Sigma.\mathsf{sk}, c)$ and return (c, π) to $\mathcal{A}$.
- When $\mathcal{A}$ issues a vCCA^D game decryption request for ciphertext (c, π), $\mathcal{B}$ proceeds as follows:
  - If $\Sigma.\mathsf{Verify}(\Sigma.\mathsf{pk}, c, \pi) = \mathsf{True}$ (i.e., when (c, π) is a fresh well-formed ciphertext) $\mathcal{B}$ issues a CPA^D game decryption request on $\mathsf{idx}(c)$ and return the result to $\mathcal{A}$.
  - If $\Pi.\mathsf{Verify}([\Pi.\mathsf{vk}], c, \pi) = \mathsf{True}$, $\mathcal{B}$ invokes Π's $\mathsf{Extract}$ procedure over $(c; \pi)$ to get $f; (c_1; \pi_1), ..., (c_K; \pi_K)$. In this case, if $\Sigma.\mathsf{Verify}(\Sigma.\mathsf{pk}, c_i, \pi_i)) = \mathsf{False}$ for some i $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$. Otherwise (when (c, π) is a well-formed evaluated ciphertext), $\mathcal{B}$ then does a CPA^D game evaluation request with parameters $f; \mathsf{idx}(c_1), ..., \mathsf{idx}(c_K)$ and get ciphertext $c' = c$ (recall that Eval is deterministic) in return (with also the effect of adding $c' = c$ along with the associated left and right cleartext evaluations in the CPA^D game state), $\mathcal{B}$ also performs the left and right cleartext evaluations for itself to get
$$m'_0 = f(S^{\mathcal{B}}[\mathsf{idx}(c_1)].m_0, ..., S^{\mathcal{B}}[\mathsf{idx}(c_K)].m_0)$$
and
$$m'_1 = f(S^{\mathcal{B}}[\mathsf{idx}(c_1)].m_1, ..., S^{\mathcal{B}}[\mathsf{idx}(c_K)].m_1)$$
and do $S^{\mathcal{B}} := [S^{\mathcal{B}}; (m'_0, m'_1, c')]$. Finally, $\mathcal{B}$ issues a CPA^D game decryption request with $\mathsf{idx}(c') = |S^{\mathcal{B}}|$ and returns the result to $\mathcal{A}$.
  - Otherwise, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.

Thus, $\mathcal{B}$ can duly simulate all the vCCA^D game requests issued by $\mathcal{A}$. And eventually, $\mathcal{B}$ forwards the decision of $\mathcal{A}$: when $\mathcal{A}$ wins, $\mathcal{B}$ wins. □

Remark that in the above proof, the simulator $\mathcal{B}$ does not provide an evaluation oracle and, hence, does not generate any proofs: $\mathcal{B}$ only invokes the SNARK extractor when $\Pi.\mathsf{Verify}(\pi) = \mathsf{True}$ on a proof π it has not generated. So $\mathcal{A}$ has only access to proofs it has generated by itself and, as such, has no access to any simulator oracle against Π. It follows that the straightline-extractability of Π is sufficient for the vCCA^D security of the construction to hold without any additional non-malleability property for the SNARK.

6.2 Public-key constructions

We now consider the public key, designated-verifier[14] construction blueprint proposed in [MN24] which we refer to as the *CCA2-Companion-Ciphertext* approach in this paper. The construction is built over a public-key FHE scheme $\mathcal{E}_H$, a public-key (CCA2-secure) scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, and a publicly verifiable or designated verifier SNARK $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for langage
$$L_2 = \{c_e, c_1, ..., c_K | \exists f \in \mathcal{F}_H, c_e = \mathcal{E}_H.\mathsf{Eval}(\mathcal{E}_H.\mathsf{pk}, f, c_1, ..., c_K)\}, \tag{15}$$
to obtain encryption scheme $\mathcal{E}^{\star}_H$:

[14] In [MN24] terminology this just tells whether or not the well-formedness of fresh ciphertexts is verifiable publicly or privately. This is independent of whether the SNARK is publicly verifiable or designated-verifier.

- $\mathcal{E}_H^\star$.KeyGen: run $\mathcal{E}_H$.KeyGen, $\mathcal{E}$.KeyGen, Π.Setup, let $\mathsf{ek} = (\mathcal{E}_H.\mathsf{pk}, \mathcal{E}.\mathsf{pk})$ as well as $\mathsf{sk} = (\mathcal{E}_H.\mathsf{sk}, \mathcal{E}.\mathsf{sk}, [\Pi.\mathsf{vk}])$.
- $\mathcal{E}_H^\star$.Enc: given $m \in \mathcal{P}$ generate ciphertext, with $|$ denoting the concatenation operator, $(c, \pi) = (\mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk}, m; r), \mathcal{E}.\mathsf{Enc}(\mathcal{E}.\mathsf{pk}, m|r))$ (the CCA2 companion ciphertext is denoted by π as its intent is to act as a proof of knowledge, see below).
- $\mathcal{E}_H^\star$.Eval: given ciphertexts $(c_1, \pi_1), ..., (c_K, \pi_K)$, compute

$$(c_e = \mathcal{E}_H.\mathsf{Eval}(\mathcal{E}_H.\mathsf{pk}, f, c_1, ..., c_K), \pi_e = \Pi.\mathsf{Prove}((c_e, c_1, ..., c_K), f)).$$

  and return ciphertext $((c_e, \pi_e), (c_1, \pi_1), ..., (c_K, \pi_K))$.
- $\mathcal{E}_H^\star$.Dec (fresh ciphertext): given (c, π), unless $\mathsf{Verif}(\mathcal{E}.\mathsf{sk}, c, \pi) = \mathsf{True}$ return $\perp$ and otherwise return $\mathcal{E}_H.\mathsf{Dec}(\mathcal{E}_H.\mathsf{sk}, c)$[15].
- $\mathcal{E}_H^\star$.Dec (evaluated ciphertext): given ciphertext $((c_e, \pi_e), (c_1, \pi_1), ..., (c_K, \pi_K))$, if $\mathsf{Verif}(\mathcal{E}.\mathsf{sk}, c_i, \pi_i) = \mathsf{True}, \forall i$ and $\Pi.\mathsf{Verify}([\Pi.\mathsf{vk}], c_e, \pi_e)$, return $\mathcal{E}_H.\mathsf{Dec}(\mathcal{E}_H.\mathsf{sk}, c_e)$. Return $\perp$ otherwise.

with $\mathsf{Verif}(\mathcal{E}.\mathsf{sk}, c, \pi)$ running $(m', r') = \mathcal{E}.\mathsf{Dec}(\mathcal{E}.sk, \pi)$ and returning True if and only if

$$c = \mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk}, m'; r').$$

Intuitively, the essence of this construction is to define fresh ciphertexts as the association of an FHE ciphertext encrypting m by means of randomness r and another ciphertext encrypting the concatenation of m and r under a CCA2-secure encryption scheme, as a proof of knowledge. This indeed allows to verify the well-formedness of these fresh ciphertexts by first decrypting the companion CCA2 ciphertext to recover m and r[16] and then checking that the associated FHE ciphertext is indeed equal to $\mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk}, m; r)$. Note that the verification may succeed when $\mathcal{E}_H.\mathsf{Dec}(\mathcal{E}_H.\mathsf{sk}, \mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk}, m; r)) \neq m$ which is what we want when the correctness assumption does not hold for $\mathcal{E}_H$.

Equivalently, following [MN24], we may consider the verification performed by means of $\mathcal{E}$ as a *designated-verifier* proof of knowkedge $\Pi_\mathcal{E}$ for language

$$L_3 = \{c | \exists m \in \mathcal{P}, \exists r \in \mathsf{COIN}, c = \mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk}, m; r)\}, \tag{16}$$

with the following functions:

- $\Pi_\mathcal{E}$.Setup: $\mathcal{E}$.KeyGen.
- $\Pi_\mathcal{E}.\mathsf{Prove}(c, (m, r))$: returns $\pi = \mathcal{E}.\mathsf{Enc}(m|r)$.
- $\Pi_\mathcal{E}.\mathsf{Verify}(c, \pi)$: let $m'|r' = \mathcal{E}.\mathsf{Dec}(\mathcal{E}.\mathsf{sk}, \pi)$, return True if $c = \mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk}, m'; r')$ and False otherwise.

We then have the following Lemma which we prove in Appendix B.5.

Lemma 3. If $\mathcal{E}$ is a public-key CCA2-secure encryption scheme, $\Pi_\mathcal{E}$ is a designated-verifier proof of knowledge for language L_3 (Eq. 16*), with zero-knowledge property and straightline extractability.*

[15] Here, we slightly depart from the construction of [MN24] in the following sense. When decrypting a fresh ciphertext (c, π), they indeed proceed by calling $\mathcal{E}.\mathsf{Dec}(\mathcal{E}.\mathsf{sk}, \pi)$ to get m and r and return m when $\mathsf{Verif}(\mathcal{E}.\mathsf{sk}, c, \pi) = \mathsf{True}$ (i.e., they never decrypt the FHE ciphertext). We, on the contrary, return $\mathcal{E}_H.\mathsf{Dec}(\mathcal{E}_H.\mathsf{sk}, c)$ when $\mathsf{Verif}(\mathcal{E}.\mathsf{sk}, c, \pi) = \mathsf{True}$. Although both options are equivalent under the correctness assumption of $\mathcal{E}_H$, this is not the case in the general regime. However, when $\mathcal{E}_H$ is CPA^D-secure (as required for the construction in the general regime), this difference has no security implications.

[16] As such, in the CCA2-Companion-Ciphertext construction, we exactly get the additional straigthline-extractor $\mathsf{Extract}'$ needed in the vCCA^D game definition in the public key case (Sect. 3.3).

The approach, however, has the drawback that it cannot achieve any form of input privacy as this verification requires the knowledge of the CCA2 scheme decryption key and, as a consequence, can be performed only in the decryption function of the overall scheme, requiring the availability of the input ciphertexts. This also makes it non compact.

The above CCA2-Companion-Ciphertext blueprint was proved in [MN24] to lead a vCCA_{SC}-secure scheme from a CPA-secure *correct* FHE scheme, a CCA2-secure scheme and a simulation-extractable SNARK (which implies the existence of an extractor as defined in Sect. 3.2). Their proof technique consists in showing that a successful vCCA_{SC} attack over scheme $\mathcal{E}_H^\star$ implies a successful CCA2 attack on the companion CCA2 scheme. As in the previous section, it turns out that the security of this construction goes beyond this setting as we now prove that it also achieves vCCA^D security in the general regime, as long as we instantiate it from a (strong) CPA^D secure rather than, as in [MN24], a CPA-secure/correct FHE. The notion of strong CPA^D (sCPA^D) security has recently been introduced in [BJSW24] with the main difference that, in the strong CPA^D game, the adversary further controls the randomness that serves to create an otherwise well-formed ciphertext from a message m. That paper also shows that sCPA^D is strictly stronger than CPA^D. Taking a glimpse at the proof of the next proposition, it turns out that in the last reduction (when $\mathcal{B}$ operates both $\mathcal{E}_H$ and Π), the fresh ciphertexts generated by $\mathcal{A}$ must end up in the challenger's internal state (then $\mathcal{B}$ can make the latter perform the evaluation request which will also put the very same evaluated ciphertext that $\mathcal{A}$ is asking for to decrypt within the challenger state). For this reason, we indeed need to communicate the randomness to the encryption oracle of the challenger which is therefore exactly a sCPA^D rather than a CPA^D challenger.

In a nutshell, the next proof proceeds with two steps of game hopping, respectively relying on the CCA2 security of $\mathcal{E}$ (or, equivalently, the straightline-extractability of $\Pi_{\mathcal{E}}$ above) and the straightline-extractability of Π_1, followed by a final reduction towards the CPA^D security of the underlying FHE scheme.

Proposition 17 (CCA2-Companion-Ciphertext)**.** *Let $\mathcal{A}$ be an adversary against the vCCA^D security of $\mathcal{E}_H^\star$, then, under the assumption that $\mathcal{E}$ is CCA2-secure and Π is straightline-extractable, there exists an adversary $\mathcal{B}$ against the sCPA^D security of $\mathcal{E}_H$ which uses $\mathcal{A}$ as a subroutine.*

Proof. Let us start with the game G_0 which is the vCCA^D game.

First game hop. Let us now consider the game G_1 which is the same as G_0 except that the challenger call a verification oracle instead of verifying by itself proofs under $\Pi_{\mathcal{E}}$. It then follows from the the knowledge soundness of proof of knowledge $\Pi_{\mathcal{E}}$ (Lemma 3), that G_0 and G_1 are exactly the same. Indeed, the extractor used in the verification oracle operates exactly as $\Pi_{\mathcal{E}}.\mathsf{Verify}$ does, since $\Pi_{\mathcal{E}}.\mathsf{Verify}$ already extract the witness of the proof. Hence,

$$|P(O_0) - P(O_1)| = 0$$

Second game hop. As in the proof of Proposition 16 we now consider game G_2, which is the same as G_1 except that the challenger replaces all verifications of the proofs under Π by invoking a verification oracle and then the straightline extractor to get the witnesses it checks by itself. This is the same second step as in the proof of Proposition 16.

Final reduction. The end of the proof is quite similar to that of Proposition 16 except that we slightly modify the CPA^D encryption oracle (but *not* the challenge oracle) such that it further takes randomness r as a parameter. This modification exactly fits the definition of Strong CPA^D introduced in [BJSW24] and is thus the reason why we now show that an adversary $\mathcal{B}$ against the sCPA^D (rather than CPA^D) security of $\mathcal{E}_H$ can be built using $\mathcal{A}$ against G_2 (or equivalently G_0) as a subroutine. Then, $\mathcal{B}$ initially runs $\Pi.\mathsf{Setup}$ and $\Pi_{\mathcal{E}}.\mathsf{Setup}$ and communicates the associated public material to $\mathcal{A}$ (which does not contain the verification key of the designated-verifier proof $\Pi_{\mathcal{E}}$). When $\mathcal{B}$ receives a challenge request

with $m_0 \neq m_1$, it forwards it to the sCPA^D challenger to get $c = \mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk}, m_b; r)$ with unknown b and r, $\mathcal{B}$ then updates $S^{\mathcal{B}} := [S^{\mathcal{B}}; (m_0, m_1, c)]$ and returns (c, π') for a simulated proof π', that is indistinguishable from a real proof for $\mathcal{A}$, without the verification material, under the zero-knowledge property (of a designated-verifier proof system). When $\mathcal{B}$ processes a decryption request (assuming evaluated ciphertexts), $\mathcal{B}$ retrieves the input ciphertexts using $\Pi.\mathsf{Extract}$. The input ciphertexts which are challenge ones are already in his or her internal state (ditto for the sCPA^D challenger). For the input ciphertexts which are challenge-independent, the message and randomness are recovered by $\mathcal{B}$ via the straightline-extractor $\Pi_{\mathcal{E}}.\mathsf{Extract}$, which thus implements the $\mathsf{Extract}'$ defined in Sect. 3.3. Then $\mathcal{B}$ can issue the proper sCPA^D encryption requests (with the additional randomness parameter) to populate the sCPA^D challenger state (and his or her mirrored one at the same time). Then $\mathcal{B}$ can issue the sCPA^D evaluation request with the appropriate game state indices to obtain $c' = c$ (due to the deterministic evaluation assumption) and add it to the sCPA^D challenger game state. Finally, $\mathcal{B}$ issues the sCPA^D decryption request with $\mathsf{idx}(c)$. □

Lastly, let us emphasize that the (compact) Naor-Yung-based [NY90] construction of [MN24], which, in order to encrypt a plaintext, associates two FHE ciphertexts of this plaintext under different keys, and bind them by a proof that these two ciphertexts are encrypting the same plaintext, requires *perfect* correctness. Indeed, in [NY90] (definition 3.4), the scheme used in the construction must verify $\forall m \in \mathcal{P}, \forall r \in \{0;1\}^{p(n)}, \mathsf{Dec}(\mathsf{Enc}(m, r)) = m$, and this strong property lies at the heart of the validity of the proof in [NY90], see also [DNR04]. As such it is not applicable in the general case where approximate FHE schemes are allowed.

Still, vCCA^D-secure constructions achieving compactness in the public-key setting can be obtained by replacing the CCA2-companion ciphertext (which essentially provides a designated-verifier proof of plaintext awareness) by a *publicly-verifiable* zk–SNARK Π_0 for language L_3 (Eq. 16). This leads to a new Encrypt-then-Prove construction blueprint which is *identical* to the first Encrypt-then-Sign blueprint in previous Sect. 6.1, but with the signature scheme Σ replaced by Π_0 and the SNARK Π replaced by a SNARK Π_1 (which can be either publicly verifiable or designated verifier) for the following language

$$L_4 = \left\{ c_e \middle| \exists f \in \mathcal{F}_H, \exists (c_1, \pi_1), ..., (c_K, \pi_K) \in \mathcal{C}^K, \begin{array}{l} \Pi_0.\mathsf{Verify}(c_i, \pi_i), \forall i \\ c_e = \mathcal{E}_H.\mathsf{Eval}(\mathcal{E}_H.\mathsf{pk}, f, c_1, ..., c_K) \end{array} \right\}.$$

Π_0 has to be publicly verifiable since Π_1's proofs are generated by the adversary which cannot be granted access to a private verification key for Π_0.

We now prove the vCCA^D security of this latter construction. For this last proof, we proceed again with two steps of game hopping, this time respectively relying on the simulation-extractability of Π_0 and the straightline-extractability of Π_1, followed by a final reduction towards the CPA^D security of the underlying FHE scheme.

Proposition 18 (Encrypt-then-Prove)**.** *Let $\mathcal{A}$ be an adversary against the vCCA^D security of $\mathcal{E}_H^\star$. Then, under the assumption that Π_0 is simulation-extractable and Π_1 is straightline-extractable, there exists an adversary $\mathcal{B}$ against the sCPA^D security of $\mathcal{E}_H$ which uses $\mathcal{A}$ as a subroutine.*

Proof. As in the proof of Proposition 17 we consider games G_0, G_1 and G_2 which respectively are the original vCCA^D game, the game in which the challenger is modified to rely on a verification oracle instead of verifying by itself proofs under Π_0 and the game in which the challenger is further modified in the same way for the verification of the proofs under Π_1. Then, G_0 is indistinguishable from G_2 following the same game-hoping arguments as in the proof of Proposition 17 (recall that we already formalized the verification performed by means of the CCA2 scheme $\mathcal{E}$ as a proof of knowledge $\Pi_{\mathcal{E}}$ for the same language as that of Π_0).

Final reduction. Finally, as a last step, there remains to show that an adversary $\mathcal{B}$ against the sCPA^D security of $\mathcal{E}_H$ can be built using an adversary $\mathcal{A}$ against G_2 (or equivalently G_0) as a subroutine. For this last step, we then proceed similarly as in the last reduction in the proof of Proposition 17 but with $\mathcal{B}$ operating both Π_0 and Π_1. The main difference is that we construct an adversary $\mathcal{B}$ against the sCPA^D security of scheme $\mathcal{E}'_H$ (rather than $\mathcal{E}_H$) which is obtained from scheme $\mathcal{E}_H$ by just modifying the encryption function such that it also generates a proof of well-formedness i.e.,

$$(c, \pi) = \mathcal{E}'_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk}, m; r) = (\mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk}, m; r), \Pi_0.\mathsf{Prove}(c, (m, r))).$$

In particular, upon decrypting a fresh ciphertext, $\mathcal{E}'_H$ decryption function ignores any proof material and for evaluated ciphertexts $\mathcal{E}'_H$ decryption function does not takes any proof as argument. Because Π_0 is zero knowledge, the sCPA^D security of $\mathcal{E}'_H$ directly follows from that of $\mathcal{E}_H$ under the simulation soundness of Π_0. Then challenge and decryption requests are processed as in the proof of Proposition 17 with the following slight differences:

- When $\mathcal{B}$ receives a challenge request with $m_0 \neq m_1$, it forwards it to the sCPA^D challenger (against $\mathcal{E}'_H$) to get $(c, \pi) = (\mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk}, m_b; r), \Pi_0.\mathsf{Prove}(c, (m_b, r)))$, updates its internal state $S^{\mathcal{B}} := [S^{\mathcal{B}}; (m_0, m_1, c)]$ and returns (c, π) to $\mathcal{A}$.
- $\mathsf{Extract}'$ (required in the simulation of $\mathcal{A}$'s decryption requests) is realized by means of Π_0 extractor which $\mathcal{B}$ can invoke.

Note that $\mathcal{B}$ only has access to the traces of execution of $\mathcal{A}$ (which forms the auxiliary data of Π_0's extractor), therefore $\mathsf{Extract}'$ is not defined for proofs generated by the sCPA^D challenger (and thus $\mathcal{B}$ cannot retrieve b by that mean). □

As a last remark, let us emphasize that Prop. 17 and 18 perform reductions towards a "CPA^D" challenger in a public-key CCA setting. In order for the reduction to properly rely on the challenger's decryption oracle to handle decryption requests from the adversary, it has to populate the challenger's internal state with the well-formed fresh ciphertexts generated on its own by the adversary (which, in that case, controls the encryption randomness). To do so, the proof hence necessarily has to operate in the adversarially-chosen encryption randomness setup which is exactly accounted by Strong CPA^D (vs "standard" CPA^D). Note that this does not apply to the private key setting, as in this case the adversary does not generate any well-formed fresh ciphertexts by itself. We think the only way whereby this reliance on Strong CPA^D could be avoided, would be by proceeding via a reduction which does not have to rely on a challenger with a decryption oracle (e.g a reduction to the CPA rather than "CPA^D" security of the FHE scheme). This however cannot be done without introducing additional assumptions about the FHE scheme as the reduction would then have to be able to handle the adversary decryption requests without relying on any decryption oracle. This may be possible with some LWE-based schemes relying on noise flooding to achieve CPA^D security (a question we leave as an open problem), but this does not appear possible in the general setting that our paper addresses.

7 Conclusion and future work

Following the work of Manulis & Nguyen [MN24] as well as the improvements on that work we presented in this paper, designing practical FHE-style malleable schemes enforcing CCA security properties beyond the CCA1 barrier seems within reach, at least for specific applications. Indeed, recent advances in SNARK for ring arithmetic, such as [GNS23], give us the necessary toolbox for attempting concrete instantiations of the construction blueprints discussed in Sect. 6. Furthermore, in many usual applications of FHE, the set

of algorithms that needs to run in the encrypted domain is very limited (for example, a FHE aggregation server involved in a Federated Training protocol for a machine learning model may only have to run a simple average [GSCS+23] or a majority voting algorithm [GSPZ+21, GSZS+23]). This gives us hope to be able to design practical vCCA^D-secure schemes with simplified SNARK or Verifiable Computing techniques tailored to these sets of algorithms. Lastly, it will also be interesting to investigate which building blocks are friendly towards each others e.g., finding "SNARK friendly" signature schemes for concrete instantiation of the Encrypt-then-Sign blueprint. Regarding the Encrypt-then-Prove blueprint, we also think that targeting applications requiring only linear operations by means of a (R)LWE-based scheme with a Regev-style public-key (so that all well-formed ciphertexts end up being linear combinations of the encryptions of 0 forming the public key) might be a good playground towards obtaining first practical instantiations of the blueprint.

Also, following a recent burst of new CPA^D attacks on both noise-flooded CKKS and "exact" schemes such as BFV, BGV or TFHE [GNSJ24, CSBB24, CCP+24] new FHE security paradigms are being proposed. As an example, Alexandru et al. [ABMP24] have proposed a new *weaker* variant of CPA^D security, termed application-aware security. In essence, this new definition acknowledges that for non-exact FHE schemes, CPA^D security should be defined relatively to a function class $\mathcal{F}_C$ and a ciphertext noise estimation strategy, rather than absolutely. With respect to that new security notion, the cryptosystem parameters should then be set relatively to these, and the homomorphic evaluations should be limited to the functions or circuits in the class. However, one of the main drawbacks of the application-aware approach is that the burden of enforcing the above constraints lies, so far, solely on the library user's shoulders (see also [BBB+22] and, in particular, its new Sect. 2.6.1). As a starting point, an interesting line of research would then be to connect the application-aware paradigm with both vCCA and vCCA^D security notions by defining new weaker variants of these notions, e.g. $\mathcal{F}_C$-vCCA and $\mathcal{F}_C$-vCCA^D security, for leveraging somewhat correct (and CPA) or CPA^D schemes, i.e. schemes achieving correctness or CPA^D security only over $\mathcal{F}_C$, to CCA security levels. For example, the vCCA$_{\mathrm{SC}}$ and vCCA^D decryption oracles may further check that $f \notin \mathcal{F}_C$ in conditions (7) and (8), respectively (which is precisely what is suggested for the CPA^D game evaluation oracle in [ABMP24]). Our intuitions are that the picture depicted in this paper will be relatively similar for these restricted security notions but we leave this for further work. However, a difficult point will be to capture the dependency of the application aware approach upon noise estimation strategies in meaningful CCA security notions, with the hope of achieving both beyond CCA1 security and relieving the library users of the burden of enforcing by hand the constraints of that paradigm.

Acknowledgments

The authors would like to thank the anonymous referees for a number of suggestions that helped improving this paper.

References

[ABMP24] A. Alexandru, A. Al Badawi, D. Micciancio, and Y. Polyakov. Application-aware approximate homomorphic encryption: Configuring FHE for practical use. Technical Report 203, IACR ePrint, 2024. URL: https://eprint.iacr.org/2024/203.

[BBB+22] A. Al Badawi, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, Z. Liu, D. Micciancio, I. Quah, Y. Polyakov,

R. V. Saraswathy, K. Rohloff, J. Saylor, D. Suponitsky, M. Triplett, V. Vaikuntanathan, and V. Zucca. OpenFHE: Open-source fully homomorphic encryption library. In *WAHC*, pages 53–63, 2022. doi:10.1145/3560827.3563379.

[BDJR97] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *IEEE SFCS*, pages 394–403, 1997. doi:10.1109/SFCS.1997.646128.

[BDPR98] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO*, pages 26–45, 1998. doi:10.1007/BFb0055718.

[BGV12] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. *ACM ITCS*, pages 309 – 325, 2012. doi:10.1145/2090236.2090262.

[BJSW24] O. Bernard, M. Joye, N. P. Smart, and M. Walter. Drifting towards better error probabilities in fully homomorphic encryption schemes. Technical Report 1718, IACR ePrint, 2024. URL: https://eprint.iacr.org/2024/1718.

[Bra12] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO*, pages 868–886, 2012. doi:10.1007/978-3-642-32009-5_50.

[CCP+24] J. H. Cheon, H. Choe, A. Passelègue, D. Stehlé, and E. Suvanto. Attacks against the IND-CPAD security of exact FHE schemes. In *CCS*, pages 2505 – 2519, 2024. doi:10.1145/3658644.3690341.

[CGGI16] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *ASIACRYPT*, pages 3–33, 2016. doi:10.1007/978-3-662-53887-6_1.

[CKKS17] J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT*, pages 409–437, 2017. doi:10.1007/978-3-319-70694-8_15.

[CSBB24] M. Checri, R. Sirdey, A. Boudguiga, and J.-P. Bultel. On the practical CPAD security of "exact" and threshold FHE schemes. In *CRYPTO*, pages 3–33, 2024. doi:10.1007/978-3-031-68382-4_1.

[DNR04] C. Dwork, M. Naor, and O. Reingold. Immunizing encryption schemes from decryption errors. In *EUROCRYPT*, pages 342–360, 2004. doi:10.1007/978-3-540-24676-3_21.

[FV12] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. Technical Report 2012/144, IACR ePrint, 2012. URL: https://eprint.iacr.org/2012/144.

[GNS23] C. Ganesh, A. Nitulescu, and E. Soria-Vazquez. Rinocchio: SNARKs for ring arithmetic. *J. Cryptol.*, page 41, 2023. doi:10.1007/s00145-023-09481-3.

[GNSJ24] Q. Guo, D. Nabokov, E. Suvanto, and T. Johansson. Key recovery attacks on approximate Homomorphic Encryption with nonworst-case noise flooding countermeasures. In *Usenix Security*, pages 7447–7461, 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/guo-qian.

[GSCS+23] A. Grivet-Sébert, M. Checri, O. Stan, R. Sirdey, and C. Gouy-Pailler. Combining Homomorphic Encryption and differential privacy in federated learning. In *IEEE PST*, pages 1–7, 2023. doi:10.1109/PST58708.2023.10320195.

[GSPZ+21] A. Grivet-Sébert, R. Pinot, M. Zuber, C. Gouy-Pailler, and R. Sirdey. SPEED: secure, private, and efficient deep learning. *Machine Learning*, pages 675–694, 2021. doi:10.1007/s10994-021-05970-3.

[GSZS+23] A. Grivet-Sébert, M. Zuber, O. Stan, R. Sirdey, and C. Gouy-Pailler. A probabilistic design for practical homomorphic majority voting with intrinsic differential privacy. In *WAHC*, pages 47–58, 2023. doi:10.1145/3605759.3625258.

[LM21] B. Li and D. Micciancio. On the security of homomorphic encryption on approximate numbers. In *EUROCRYPT*, pages 648–677, 2021. doi:10.1007/978-3-030-77870-5_23.

[LMSS22] B. Li, D. Micciancio, M. Schultz, and J. Sorrell. Securing approximate homomorphic encryption using differential privacy. In *CRYPTO*, pages 560–589, 2022. doi:10.1007/978-3-031-15802-5_20.

[MN24] M. Manulis and J. Nguyen. Fully homomorphic encryption beyond IND-CCA1 security: Integrity through verifiability. In *EUROCRYPT*, pages 63–93, 2024. doi:10.1007/978-3-031-58723-8_3.

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *ACM STOC*, pages 427–437, 1990. doi:10.1145/100216.100273.

[VKH23] A. Viand, C. Knabenhans, and A. Hithnawi. Verifiable fully homomorphic encryption. Technical Report 2301.07041, arXiv, 2023. URL: https://arxiv.org/abs/2301.07041.

A Formal preliminaries on SNARKs

Let us recall the formal definition of (zero-knowledge) succinct non-interactive arguments of knowledge (zk-SNARKs), for a Boolean relation $\mathcal{R}$ on words or statements u and witnesses w, of an $\mathcal{NP}$ language. The $\mathcal{NP}$ language L being defined as $L = \{u | \exists w, \mathcal{R}(u, w) = \mathsf{True}\}$.

Let us remark that all the notions below define proofs of knowledge. Succintness is a special property for SNARKs. Zero-knowkedge is an independent property.

Definition 1 (SNARK)**.** A SNARK Π is defined by three algorithms,

Setup$(1^\lambda, \mathcal{R})$**:** on input 1^λ and an $\mathcal{NP}$ relation $\mathcal{R}$, the generation algorithm outputs a common reference string crs, assumed to be used in both subsequent algorithms. It can optionally output a verification key vk that is secret in the case of designater-verifier SNARK;

Prove(u, w)**:** given an instance u and a witness w such that $\mathcal{R}(u, w) = \mathsf{True}$, this algorithm produces a proof π;

Verify$([\mathsf{vk}], u, \pi)$**:** on, the optional verification key vk, an instance u, and a proof π, the verifier algorithm outputs False (reject) or True (accept);

satisfying the following properties:

Correctness. For any $u \in L$, with witness w,

$$\Pr\left[\mathcal{V}([\mathsf{vk}], u, \pi) = \mathsf{False} \,\middle|\, \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}), \pi \leftarrow \mathsf{Prove}(u, w)\right] \leq \mathrm{neg}(\lambda);$$

Succintness. The size of the proof is linear in the security parameter λ, i.e. independent of the size of the computation or the witness;

Knowledge-Soundness. For any PPT adversary $\mathcal{A}^{\mathsf{ks}}$ there exists a PPT extractor $\mathcal{E}_{\mathcal{A}}$ such that:

$$\Pr\left[\begin{array}{c} \mathsf{Verify}([\mathsf{vk}], u, \pi) = \mathsf{True} \\ \wedge\ \mathcal{R}(u, w) = \mathsf{False} \end{array} \,\middle|\, \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}) \\ ((u, \pi), \mathsf{aux}) \leftarrow \mathcal{A}^{\mathsf{ks}}(\mathsf{crs}) \\ w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{crs}, (u, \pi), \mathsf{aux}) \end{array}\right] \leq \mathrm{neg}(\lambda).$$

Intuitively, this means that for any prover able to produce a valid proof π for a statement u in the language, there exists an efficient extractor that outputs a witness w for the given statement u, from a trace aux of the execution of the adversary;
In case of Designated-Verifier SNARK, the adversary is given access to a verification oracle.

Zero Knowledge. There exists a stateful interactive polynomial-size simulator $\mathsf{Sim} = (\mathsf{Simcrs}, \mathsf{SimProve})$ such that for all stateful interactive distinguishers $\mathcal{D}$, the two probabilities are negligibly close:

$$\begin{aligned} &\Pr[\mathcal{R}(u, w) = \mathsf{True} \wedge \mathcal{D}(\pi) = 1 \,|\, \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}), (u, w) \leftarrow \mathcal{D}(\mathsf{crs}), \\ &\qquad\qquad \pi \leftarrow \mathsf{Prove}(u, w)]; \\ &\Pr[\mathcal{R}(u, w) = \mathsf{True} \wedge \mathcal{D}(\pi) = 1 \,|\, (\mathsf{crs}, \mathsf{trap}) \leftarrow \mathsf{Simcrs}(1^\lambda), (u, w) \leftarrow \mathcal{D}(\mathsf{crs}), \\ &\qquad\qquad \pi \leftarrow \mathsf{SimProve}(\mathsf{crs}, \mathsf{trap}, u)]. \end{aligned}$$

In case of Designated-Verifier SNARK, the distinguisher is just given the crs and not the verification key.

We stress that the above notation of extractor $\mathcal{E}_{\mathcal{A}}$ limits to a *straightline extractability* (without possible rewinding), but this is what we will need.

B Additional proofs

B.1 Proof of proposition 4 ($\mathrm{vCCA}_{\mathrm{SC}} \not\Longrightarrow \mathrm{CPA}_2^D$, p. 13)

Proof. We proceed similarly to the proof of Proposition 2. Let us start from a $\mathrm{vCCA}_{\mathrm{SC}}$-secure scheme $S = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ from which we build the scheme S' with the only modification that

$$\mathsf{Enc}'(m) = \mathsf{Enc}(m + g(m)).$$

with g as in the proof of Proposition 2.
S' is $vCCA_{\mathrm{SC}}$-secure. Let $\mathcal{A}$ be a successful adversary against the $\mathrm{vCCA}_{\mathrm{SC}}$ security of S'. It is then easy to build an adversary $\mathcal{B}$ against the $\mathrm{vCCA}_{\mathrm{SC}}$ security of S. Indeed, $\mathcal{B}$ simulates $\mathcal{A}$ encryption (respectively challenge) requests simply by forwarding $m + g(m)$ (respectively $m_0 + g(m_0)$ and $m_1 + g(m_1)$) as an encryption (respectively challenge) request to the $\mathrm{vCCA}_{\mathrm{SC}}$ challenger against S. All other requests are transferred "as is" by $\mathcal{B}$ to the $\mathrm{vCCA}_{\mathrm{SC}}$ game against S.
S' is CPA_2^D-insecure. Identical to proof of Proposition 2: the CPA_2^D decryption oracle accepts the c_{mul} ciphertext since it is duly registered in the game state within the triplet $(0, 0, c_{\mathrm{mul}})$ as an encryption of 0 with respect to S'. □

B.2 Proof of proposition 5 ($\mathbf{CPA}_2^D \not\Longrightarrow \mathbf{vCCA}_{\mathrm{SC}}$, p. 13)

Proof. The proof is essentially identical to that of Proposition 1, but starting from a CPA_2^D-secure scheme S from which we create a scheme S' in a similar way. On one hand, the CPA_2^D-security of S' follows from that of S and the fact that a CPA_2^D adversary against S' cannot add $c^{\triangle}$ to the game state. On the other hand, the $\mathrm{vCCA}_{\mathrm{SC}}$-insecurity of S' follows from the fact that the $\mathrm{vCCA}_{\mathrm{SC}}$ game decryption oracle accepts $c^{\triangle}$ as it bears no relationship with the challenge ciphertext. □

B.3 Proof of proposition 8 ($\mathbf{vCCA}_{\mathrm{SC}} \not\Longrightarrow \mathbf{vCCA}_{\mathbf{SC}}^D$, p. 15)

Proof. We proceed similarly to the proof of Proposition 2 (and Proposition 4). Let us start from a $\mathrm{vCCA}_{\mathrm{SC}}^D$-secure scheme $S = (\mathsf{KeyGen}, \mathsf{EncDec}, \mathsf{Eval})$ from which we build the scheme S' with the only modification that

$$\mathsf{Enc}'(m) = \mathsf{Enc}(m + g(m)).$$

with function g as in the proof of Proposition 2.

S' is $vCCA_{\mathrm{SC}}$-secure. Since S is $\mathrm{vCCA}_{\mathrm{SC}}^D$-secure, it is also $\mathrm{vCCA}_{\mathrm{SC}}$-secure (from Lemma 2). Now, let $\mathcal{A}$ be a successful adversary against the $\mathrm{vCCA}_{\mathrm{SC}}$ security of S'. It is then easy to build an adversary $\mathcal{B}$ against the $\mathrm{vCCA}_{\mathrm{SC}}$ security of S. Indeed, $\mathcal{B}$ simulates $\mathcal{A}$ encryption (respectively challenge) requests simply by forwarding $m + g(m)$ (respectively $m_0 + g(m_0)$ and $m_1 + g(m_1)$) as an encryption (respectively challenge) request to the $\mathrm{vCCA}_{\mathrm{SC}}$ challenger against S. All other request are transferred "as is" by $\mathcal{B}$ to the $\mathrm{vCCA}_{\mathrm{SC}}$ game against S.

S' is $vCCA^D$-insecure. Identical to proof of Proposition 2: the $\mathrm{vCCA}_{\mathrm{SC}}^D$ decryption oracle accepts the c_{mul} ciphertext as, recall (9) and (10), $\mathrm{left}(c_{\mathrm{mul}}) = \mathrm{right}(c_{\mathrm{mul}}) = 0$ (as c_{mul} is an encryption of 0 with respect to S').

Since this latter attack involves only legit ciphertexts, it can be performed by an adversary working under the well-formedness assumption. □

B.4 Proof of proposition 14 (p. 19)

Proof. We proceed similarly to the proof of Proposition 10. So let us start with a vCCA^D-secure *additive* HE scheme $S = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ from which we build the scheme S' with the only modification that

$$\mathsf{Enc}'(m) = \mathsf{Enc}(m + g(m)),$$

where g is as in the proof of Proposition 10.

S' is $vCCA_{SC}^D$-secure. Since vCCA^D security trivially implies $\mathrm{vCCA}_{\mathrm{SC}}^D$ security, S is $\mathrm{vCCA}_{\mathrm{SC}}^D$-secure. Let $\mathcal{A}$ be a successful adversary against the $\mathrm{vCCA}_{\mathrm{SC}}^D$ security of S'. It is then easy to build an adversary $\mathcal{B}$ against the $\mathrm{vCCA}_{\mathrm{SC}}^D$ security of S. Indeed, $\mathcal{B}$ simulates $\mathcal{A}$ encryption and challenge requests simply by adding $g(m)$ to m. All other request are transferred "as is" by $\mathcal{B}$ to the $\mathrm{vCCA}_{\mathrm{SC}}^D$ game against S.

S' is $vCCA^D$-insecure. Identical to proof of Proposition 10: the vCCA^D decryption oracle accepts the c_{sum} ciphertext as, recall (9) and (10), $\mathrm{left}(c_{\mathrm{sum}}) = \mathrm{right}(c_{\mathrm{sum}}) = Z$ (as an encryption of Z with respect to S'). □

B.5 Proof of Lemma 3 (p. 23)

Proof. **Correctness.** For any $c \in L_3$, with witness $(m|r)$, we are interested in the following probability,

$$\Pr\left[\Pi_{\mathcal{E}}.\mathsf{Verify}([\mathsf{vk}], c, \pi) = \mathsf{False} \,|\, \pi = \Pi_{\mathcal{E}}.\mathsf{Prove}(c, (m|r))\right]$$

which can be rewritten as ($\mathcal{E}$.pk, $\mathcal{E}$.sk, $\mathcal{E}_H$.pk and $\mathcal{E}_H$.sk omitted for compacity-sake),

$$\Pr\left[\begin{array}{c|c} \mathcal{E}.\mathsf{Dec}(\pi) = (m'|r') \neq (m|r) \ \wedge \ \mathcal{E}_H.\mathsf{Enc}(m',r') \neq c \\ \vee \\ \mathcal{E}.\mathsf{Dec}(\pi) = (m|r) \ \wedge \ \mathcal{E}_H.\mathsf{Enc}(m,r) \neq c \end{array} \middle| \begin{array}{c} \pi = \mathcal{E}.\mathsf{Enc}((m|r)) \\ c = \mathcal{E}_H.\mathsf{Enc}(m,r) \end{array}\right].$$

Since the two events in the above left-hand side disjunction are incompatible, we can separate the above as,

$$\Pr\left[\begin{array}{c} \mathcal{E}.\mathsf{Dec}(\mathcal{E}.\mathsf{sk},\pi) = (m'|r') \neq (m|r) \\ \wedge \ \mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk},m',r') \neq c \end{array} \middle| \begin{array}{c} \pi = \mathcal{E}.\mathsf{Enc}(\mathcal{E}.\mathsf{pk},(m|r)) \\ c = \mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk},m,r) \end{array}\right]$$

$$+$$

$$\Pr\left[\begin{array}{c} \mathcal{E}.\mathsf{Dec}(\mathcal{E}.\mathsf{sk},\pi) = (m|r) \\ \wedge \ \mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk},m,r) \neq c \end{array} \middle| \begin{array}{c} \pi = \mathcal{E}.\mathsf{Enc}(\mathcal{E}.\mathsf{pk},(m|r)) \\ c = \mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk},m,r) \end{array}\right].$$

As $\mathcal{E}_H$.Enc is deterministic, the second of the two above terms is 0. We are then left with the first term which is bounded by,

$$\Pr\left[\begin{array}{c} \mathcal{E}.\mathsf{Dec}(\mathcal{E}.\mathsf{sk},\pi) = (m'|r') \neq (m|r) \end{array} \middle| \begin{array}{c} \pi = \mathcal{E}.\mathsf{Enc}(\mathcal{E}.\mathsf{pk},(m|r)) \\ c = \mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk},m,r) \end{array}\right] \leq \mathrm{neg}(\lambda),$$

under the (mild) assumption that $\mathcal{E}$ is statistically correct, following Eq. (1).

Knowledge-Soundness. For any PPT adversary $\mathcal{A}^{\mathsf{ks}}$ there exists a PPT extractor $\mathcal{E}_{\mathcal{A}}$ such that:

$$\Pr\left[\begin{array}{c} \mathsf{Verify}([\mathsf{vk}],c,\pi) = \mathsf{True} \\ \wedge \\ \mathcal{R}(c,w) = \mathsf{False} \end{array} \middle| \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda,\mathcal{R}) \\ ((c,\pi),\mathsf{aux}) \leftarrow \mathcal{A}^{\mathsf{ks}}(\mathsf{crs}) \\ w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{crs},(c,\pi),\mathsf{aux}) \end{array}\right] \leq \mathrm{neg}(\lambda).$$

In the present case, $\mathcal{E}_{\mathcal{A}}$ is exactly $\mathcal{E}$.Dec() (i.e., the extractor is independent of any auxiliary data and is explicitly invoked by Verify). Hence, the above probability is

$$\Pr\left[\begin{array}{c} \mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk},m',r') = c \\ \wedge \\ \mathcal{E}_H.\mathsf{Enc}(\mathcal{E}_H.\mathsf{pk},m',r') \neq c \end{array} \middle| \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda,\mathcal{R}) \\ (c,\pi) := ((c,\pi),\mathsf{aux}) \leftarrow \mathcal{A}^{\mathsf{ks}}(\mathsf{crs}) \\ (m'|r') := \mathcal{E}.\mathsf{Dec}(\mathcal{E}.\mathsf{sk},\pi) \end{array}\right] = 0$$

As $\mathcal{E}_{\mathcal{A}}$ is $\mathcal{E}$.Dec(), this is a straightline extractor.

Zero-Knowledge. The simulator can simply generate a ciphertext for a random plaintext, which will be indistinguishable from the correct ciphertext, under the indistinguishability of $\mathcal{E}$. In addition, this indistinguishability still holds with a verification oracle access, that would be simulated by a call to the decryption oracle, thanks to the CCA2 security.

□

C Additional blueprints

To improve the practicality of the Encrypt-then-Sign construction of Sect. 6.1 (at the expense of input privacy and compactness), it is also possible to modify it such that the statements for which the Π outputs a proof during Eval does not have to include the verification of the input ciphertexts' signatures. In that case, both the input *and* output ciphertexts must be available to the decryption algorithm which is then responsible for verifying the signatures of the former. When this is the case, Π is for the simpler language L_2 (i.e., (15) instead of (14)) and Eval and Dec are modified as follows:

- $\mathcal{E}^{\star}_H.\mathsf{Eval}$: given ciphertexts $(c_1, \pi_1), ..., (c_K, \pi_K)$ compute

$$(c_e = \mathcal{E}_H.\mathsf{Eval}(\mathcal{E}_H.\mathsf{pk}, f, c_1, ..., c_K), \pi_e = \Pi.\mathsf{Prove}((c_e, c_1, ..., c_K), f)).$$

  and return ciphertext $((c_e, \pi_e), (c_1, \pi_1), ..., (c_K, \pi_K))$.

- $\mathcal{E}^{\star}_H.\mathsf{Dec}$ (fresh ciphertext): given (c, π), if $\Sigma.\mathsf{Verify}(\Sigma.\mathsf{pk}, c, \pi) = \mathsf{True}$ return $\mathcal{E}_H.\mathsf{Dec}(\mathcal{E}_H.\mathsf{sk}, c)$, and return $\perp$ otherwise.

- $\mathcal{E}^{\star}_H.\mathsf{Dec}$ (evaluated ciphertext): given $((c_e, \pi_e), (c_1, \pi_1), ..., (c_K, \pi_K))$ return $\mathcal{E}_H.\mathsf{Dec}(\mathcal{E}_H.\mathsf{sk}, c_e)$ when $\Pi.\mathsf{Verify}([\Pi.\mathsf{vk}], c_e, \pi_e) = \mathsf{True}$ and $\Sigma.\mathsf{Verify}(\Sigma.\mathsf{pk}, c_i, \pi_i) = \mathsf{True}, \forall i$ and $\perp$ otherwise.

In this modified construction, the signature scheme can be replaced by a MAC $M = (\mathsf{KeyGen}, \mathsf{Tag}, \mathsf{Verify})$, leading to the Encrypt-then-MAC blueprint studied in [MN24] from a vCCA perspective, and initially proposed and proved CCA1-secure under a slightly weaker variant of (usual) CCA1 security in [VKH23].

D Separation between (q, ℓ)-CPA^D-security and $(q, \ell{+}1)$-CPA^D-security

Proposition 19. *Let ℓ be any polynomial in the security parameter. If there exists a scheme S which achieves $(q; \ell)$-IND-CPA^D-security, then there exists a scheme S' achieving $(q; \ell)$-IND-CPA^D-security, but not $(q; \ell + 1)$-IND-CPA^D-security.*

Proof. Analogously to the proof of Proposition 1, let

$$S = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}).$$

be an approximate FHE scheme that is $(q; \ell)$-IND-CPA^D-secure and achieves ε-correctness for $\varepsilon = 1$ (recall definition (2) on page 6). For simplicity sake, we do this proof under the mild assumption that the plaintext domain of S is $\mathbb{Z}_q$, for a large enough modulus q. Let $a_0, .., a_\ell, a'_0, .., a'_\ell$ be $2\ell + 2$ pairwise distinct values in the domain such that $a_0 = 1$. We define a scheme $S' = (\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Eval}')$ where encryptions, once again, consist of 2 ciphertexts where the 2nd ciphertext occasionally encrypts noise, but this time only for a_0:

$$\mathsf{Enc}'(m) = (\mathsf{Enc}(a_0), \mathsf{Enc}(3)) \text{ if } m = a_0 = 1$$
$$\mathsf{Enc}'(m) = (\mathsf{Enc}(m), \mathsf{Enc}(0)) \text{ else}$$

and let h be the function $h(m_0, m_1) = m_0 + m_0 \cdot m_1$, then we define decryption first homomorphically evaluating h and then decrypting:

$$\mathsf{Dec}'(c) = \mathsf{Dec}(\mathsf{Eval}(h, c_0, c_1)). \tag{17}$$

Note that the reason why we use the above rather than the more direct decryption function

$$\mathsf{Dec}'(c) = h\left(\mathsf{Dec}(c_0), \mathsf{Dec}(c_1)\right).$$

will become clear in the reduction at the end of the proof where it is allowed to submit the output of $\mathsf{Eval}(h, c_0, c_1)$ to the decryption oracle of a challenger against S but not the two ciphertexts c_0 and c_1, in Eq. (17), separately. For ciphertexts c honestly generated by Enc, Dec' achieves ε'-correctness for $\varepsilon' = 4$, because we increase the noise for encryptions of a_0 by $3 \cdot a_0 = 3$. Next, let f^* be the function

$$f^*(x_0, .., x_\ell) = (x_0 = a_0) \cdot .. \cdot (x_\ell = a_\ell) + (x_0 = a'_0) \cdot .. \cdot (x_\ell = a'_\ell),$$

where the equality check $(y = z)$ returns 1 iff y and z are equal. So, f^* returns 1 iff $(x_0, .., x_\ell) = (a_0, .., a_\ell)$ or $(x_0, .., x_\ell) = (a'_0, .., a'_\ell)$. Having defined f^*, we define Eval' to delete noise when $f \neq f^*$, i.e.,

$$\mathsf{Eval}'(f, \vec{c}) = (\mathsf{Eval}(f, \vec{c}_0), \mathsf{Enc}(0)) \text{ if } f \neq f^*,$$

where $\vec{c}_0$ denotes all the first ciphertext components of the pairs of ciphertexts in the vector $\vec{c}$. Moreover, when $f = f^*$, we keep the noise of the first entry:

$$\mathsf{Eval}'(f, \vec{c}) = (\mathsf{Eval}(f^*, \vec{c}_0), \vec{c}_{1,0}) \text{ if } f = f^*,$$

where $\vec{c}_{1,0}$ is the 2nd ciphertext component of the first ciphertext pair in $\vec{c}$. Since the 2nd component of a ciphertext can be at most 3 by definition of Enc and this norm is preserved by Eval and since f^* is a function with binary output, S' achieves ε'-correctness for $\varepsilon' = 4$.

Attacking $(q; \ell + 1)$-IND-CPA^D-security. An adversary against $(q; \ell + 1)$-IND-CPA^D-security of S' can now query

$$(\texttt{test messages}, a_i, a'_i)$$

for all $i \in \{0, .., \ell\}$, receiving a vector $\vec{c}$ of $\ell + 1$ ciphertext pairs. It then queries

$$(\texttt{eval}, f^*, \vec{c})$$

receiving back a pair (c_0, c_1) which it submits for decryption and receives back a value v. If $|v| \leq 1$, the adversary returns 1. If $v \geq 2$, the adversary returns 0. This decryption query is allowed because $f^*(a_0, .., a_\ell) = f^*(a'_0, .., a'_\ell) = 1$. If $b = 0$, v is a noisy encryption of 3 under S and thus ε-close to 3 for $\varepsilon = 1$. If $b = 1$, v is a noisy encryption of 0 under S and thus ε-close to 1 and thus, the adversary has advantage 1.

Proving $(q; \ell)$-IND-CPA^D-security. To prove $(q; \ell)$-IND-CPA^D-security of S', we proceed via several game hops. Game 0 is the same as the $(q; \ell)$-IND-CPA^D game for S' with secret bit $b = 0$. Game 1 is the same as Game 0, but for challenge queries even if $m_b = a_0$, adds $\mathsf{Enc}(0)$ as 2nd ciphertext component rather than $\mathsf{Enc}(3)$. Game 2 is the same as Game 1, but uses bit $b = 1$. Game 3 is the same as Game 2, but swaps back to using $\mathsf{Enc}(3)$ as 2nd ciphertext component rather than $\mathsf{Enc}(0)$ to answer challenge queries for a_0, so that Game 3 is equal to the $(q; \ell)$-IND-CPA^D-security game for S' with $b = 1$.

Each of the game-hops reduces to $(q; \ell)$-IND-CPA^D-security of S. We first focus on the most interesting game-hop from Game 1 to Game 2 and then briefly discuss the other two reductions.

Game 1 to Game 2. Let $\mathcal{A}$ be an distinguisher between Game 1 and Game 2. We construct an adversary $\mathcal{B}$ against the $(q; \ell)$-IND-CPA^D-security of S which forwards all encryption queries for messages m by $\mathcal{A}$ to its own experiment to obtain the 1st ciphertext component, and makes an encryption query for 3 to obtain the 2nd ciphertext component if $m = a_0$ and an encryption query of 0, else. Moreover, $\mathcal{B}$ forwards challenge queries by $\mathcal{A}$ for messages (m_0, m_1) as challenge queries to its own experiment to obtain the 1st part of the ciphertext and then queries 0 to its encryption oracle to obtain the 2nd ciphertext component. For evaluation queries, $\mathcal{B}$ strips off the first ciphertext components and forwards them to its own evaluation oracle to obtain the 1st ciphertext component (recall that evaluation and decryption requests should stricto sensu be made on state indices, then $\mathcal{B}$ needs to maintain a map between its own state and that of its challenger and these states are different since the CPA^D challenger's one contains the results of $\mathsf{Enc}(0), \mathsf{Enc}(3)$ and $\mathsf{Eval}(h, .)$ requests, which are unknown to $\mathcal{A}$). For the 2nd ciphertext component, if $f = f^*$, $\mathcal{B}$ keeps the noise component of the 1st ciphertext pair in the query.

If $f \neq f^*$, then $\mathcal{B}$ makes an encryption query for 0 to obtain the 2nd ciphertext component of Eval'.

For decryption queries, $\mathcal{B}$ first makes an evaluation query combining ciphertext components as Eval' would, and then makes a decryption query to its own experiment. In the end $\mathcal{B}$ returns the same bit as $\mathcal{A}$. Let's assume w.l.o.g. that $\mathcal{A}$ only makes valid decryption queries (i.e. the experiment does not return $\perp$). We now argue that all decryption queries by $\mathcal{B}$ are indeed accepted by $\mathcal{B}$'s experiment. If the 2nd component of a ciphertext (c_0, c_1) encrypts 0, then the experiment of $\mathcal{B}$ rejects if and only if the experiment of $\mathcal{A}$ rejects, because $m_0 + m_0 \cdot m_1 = m_0 + m_0 \cdot 0 = m_0$. Thus, the only difference can occur when $\mathcal{A}$ submits a ciphertext (c_0, c_1) for decryption, where neither c_0 nor c_1 are encryptions of 0.

There are two possibilities: (c_0, c_1) was the result of an encryption query of a_0 in which case decryption is allowed. Recall that even if a_0 is part of a challenge query, then the 2nd component will be an encryption of 0. Thus, the 2nd option for c_1 encrypting non-zero noise is that (c_0, c_1) was the result of an evaluation query for f^* where the first component was an encryption of a_0 (not resulting from a challenge query[17]) and the other components were encryptions of a_1,..,a_ℓ (as else, c_0 would encrypt 0). In $\mathcal{A}$'s game, such a decryption query is only allowed, if those encryptions of a_1,..,a_ℓ came from the encryption oracle, not from the challenge oracle, since swapping any of the a_i for a different value would lead f^* to evaluate to $0 \neq 1$. However, if all encryptions of a_0, a_1,..,a_ℓ come from an encryption oracle, then they are allowed in $\mathcal{B}$'s game, too.

Game 0 to Game 1. For the game hop from Game 0 to Game 1 (and analogously from Game 2 to Game 3), the reduction $\mathcal{B}$ turns all challenge queries into an encryption query to obtain the first ciphertext component, since both Game 0 and Game 1 always encrypt the left message (the right message in the game-hop from Game 2 to Game 3). For the 2nd ciphertext component, $\mathcal{B}$ makes an encryption query for 0, unless the message is a_0. In this case, $\mathcal{B}$ makes a challenge query for $(0, 3)$. Emulation of evaluation and decryption queries is similar to the previous reduction. Once more, let us assume w.l.o.g. that $\mathcal{A}$ only makes valid decryption queries and let us argue that all decryption queries by $\mathcal{B}$ are indeed allowed. Firstly, decryptions of challenge queries involving a_0 are not allowed in $\mathcal{A}$'s game, since challenge queries would be associated to two different left and right values in the challenger's state. The only other interesting case are decryption queries for results of evaluation queries on f^* where an encryption of a_0 was involved and all the other values were encryptions of a_1,..,a_ℓ. However, since $\mathcal{A}$ has only ℓ challenge queries and a_0 was already part of a challenge query, at least one of the encryptions of a_1,..,a_ℓ must come from an encryption query. In this case, however, decryptions of the result of an evaluation of f^* on these values is not allowed since swapping out one of the a_i by a different value leads f^* to evaluate to $0 \neq 1$ and the ciphertext corresponding to a_0 came from a challenge query.

Thus, S' is indeed $(q; \ell)$-IND-CPA^D-secure, but not $(q; \ell + 1)$-IND-CPA^D-secure. □

[17] If $a_0 = 1$, f^* can be applied on an output of f^* : $f^*\big(f^*(a_0, \ldots, a_\ell), a_1, \ldots, a_\ell\big)$, the 2nd component in the result of this evaluation will not be an encryption of 0.