Fully Homomorphic Encryption beyond IND-CCA1 Security: Integrity through Verifiability

Mark Manulis and Jérôme Nguyen

PACY Lab @ RI CODE, Universität der Bundeswehr München, Munich, Germany
mark@manulis.eu, jerome.nguyen@unibw.de

Abstract. We focus on the problem of constructing fully homomorphic encryption (FHE) schemes that achieve some meaningful notion of adaptive chosen-ciphertext security beyond CCA1. Towards this, we propose a new notion, called security against *verified chosen-ciphertext attack* (vCCA). The idea behind it is to ascertain integrity of the ciphertext by imposing a strong control on the evaluation algorithm. Essentially, we require that a ciphertext obtained by the use of homomorphic evaluation must be "linked" to the original input ciphertexts. We formalize the vCCA notion in two equivalent formulations; the first is in the indistinguishability paradigm, the second follows the non-malleability simulation-based approach, and is a generalization of the targeted malleability introduced by Boneh et al. in 2012.

We strengthen the credibility of our definitions by exploring relations to existing security notions for homomorphic encryption schemes, namely CCA1, RCCA, FuncCPA, CCVA, and HCCA. We prove that vCCA security is the strongest notion known so far, that can be achieved by an FHE scheme; in particular, vCCA is strictly stronger than CCA1.

Finally, we provide a general transformation, that takes *any* CPA-secure FHE scheme and makes it vCCA-secure. Our transformation first turns an FHE scheme into a CCA2-secure scheme where a part of the ciphertext retains the homomorphic properties and then extends it with a succinct non-interactive argument of knowledge (SNARK) to verifiably control the evaluation algorithm. In fact, we obtain *four* general variations of this transformation. We handle both the asymmetric and the symmetric key FHE schemes, and for each we give two variations differing in whether the ciphertext integrity can be verified publicly or requires the secret key. We use well-known techniques to achieve CCA2 security in the first step of our transformation. In the asymmetric case, we use the double encryption paradigm, and in the symmetric case, we use Encrypt-then-MAC techniques. Furthermore, our transformation also gives the *first* CCA1-secure FHE scheme based on *bootstrapping* techniques.

Keywords: Fully homomorphic encryption · Chosen-ciphertext attack · Bootstrapping · IND-vCCA · Ciphertext integrity · Verifiable FHE

1 Introduction

Fully homomorphic encryption (FHE) [37, 51] is a primitive that allows to perform computations on encrypted data without needing to first decrypt it. While the obvious application is to permit privacy-preserving outsourced computation, it has also found strong applications in other areas of cryptography such as low communication multi-party computations [46], privacy-preserving machine-learning [45] and many more, see e.g. [18, 21, 23, 39].

As is the case with traditional public key encryption schemes (PKE), the most basic security notion for FHE schemes is indistinguishability against chosen-plaintext attacks (CPA). An adversary, after receiving the public key, chooses two messages and receives a challenge ciphertext that encrypts one of them. The adversary must then decide which message has been encrypted. This needs to hold in spite of the homomorphic properties of the scheme. This is the minimal standard that an FHE scheme has to achieve. However, in stark contrast with traditional PKE where CPA is only a starting point before considering stronger security notions, CPA has so far become the standard security notion for FHE schemes (e.g. [24, 25, 38]).

This is unsatisfactory as CPA secure FHE schemes suffer from various practical attacks that arise from the lack of integrity of the ciphertexts. Observe, CPA security does not provide any guarantees if an adversary is able to access decryptions of certain ciphertexts. This may happen, for example, when a ciphertext is modified, maliciously or even accidentally. An adversary may then observe the reaction of the decrypting party to infer sensitive information. This has been leveraged by Zhang et al. [57] to construct a decryption oracle. Additionally, Akavia and Vald [2] have shown that a family of homomorphic encryption based protocols, called client-aided outsourcing protocols also suffers from attacks, even when the adversary never sees any decrypted value. Other results, e.g. [26, 43], have further shown the inadequacy of CPA secure FHE schemes for a whole host of applications.

A line of work [1,14,33,44,49] has, therefore, emerged to try to strengthen the security guarantees of FHE schemes. For traditional PKE, the "gold standard" for encryption is security against adaptive chosen-ciphertext attacks (CCA2) [28, 30,50], where the adversary additionally gets access to a decryption oracle. It can query any ciphertext apart from the actual challenge ciphertext before making its guess. Unfortunately, it is well known that homomorphic encryption schemes cannot achieve CCA2 security since homomorphism is a direct contradiction to it. Notions like generalized CCA (gCCA) [3] or replayable-CCA (RCCA) [19], which are slight relaxations of CCA2 that have been studied for PKE in the past, are similarly unachievable. Therefore, FHE schemes must be considered in the context of some weaker notion.

A first approach has been to build non-adaptive CCA (CCA1) secure FHE schemes. CCA1 is an intermediate notion, which is stronger than CPA, and where the adversary gets access to the decryption oracle *before* receiving the challenge ciphertext. Well-known group homomorphic encryption schemes such as the Pailier [48] and ElGamal [32] encryption have been proven to be CCA1 secure [4]. However, for FHE schemes the situation is more complicated.

The main breakthrough to achieve efficient FHE schemes has been the introduction of "Gentry's blueprint" [37]. This framework uses a *somewhat homomorphic encryption* (SHE) scheme and a "bootstrapping" technique to turn it into an FHE scheme. All state-of-the-art FHE schemes, e.g. [24, 25, 38], have subsequently followed this approach. However, Fauzi et al. [33] have shown that a wide array of the known SHE schemes suffer from CCA1 attacks. Additionally, a crucial requirement for bootstrapping is the need to publicly release a bootstrapping key, which is essentially an encryption of the secret key. This allows for an easy CCA1 attack, since the adversary can simply query the bootstrapping key to its decryption oracle to obtain the secret key.

To compound the issue, even CCA1-secure FHE schemes have been shown to suffer from attacks. In particular, Loftus et al. [44] show that their CCA1 construction suffers from chosen-ciphertext verification attacks (CCVA), where an adversary only gets access to an oracle that tells if a ciphertext is valid or not. This attack is, arguably, very practical. For example, in the context of outsourced computations, a server may send an evaluated ciphertext to a client and observe their reaction to deduce if the ciphertext is invalid.

Therefore, it becomes important to design FHE schemes that achieve some stronger notion than CCA1. This has previously been considered by Prabhakaran and Rosulek [49], but for a limited class of homomorphic encryption schemes supporting only univariate functions. They introduced a notion called Homomorphic-CCA (HCCA), which captures the idea that a scheme may be homomorphic for a certain set of functions but must stay non-malleable with respect to all other operations. Although HCCA is stronger than CCA1, as the adversary may access the decryption oracle after receiving the challenge ciphertext, it is not applicable to (general) FHE schemes.

Further intermediate notions between CPA and CCA2 have been considered to address specific types of attacks. Akavia et al. [1] defined a notion called FuncCPA to capture security for FHE schemes against attacks in the context of client-aided outsourcing. Boneh et al. [14] introduced targeted malleability, which similarly to HCCA, requires the scheme to stay non-malleable apart from some allowed functions. However, targeted malleability is different from HCCA in the sense that it covers FHE schemes, but only in the context of CPA and CCA1 attacks.

In light of this, we ask in this paper whether the boundaries for the security guarantees of FHE schemes can be pushed beyond CCA1? Is it possible to define a security notion that covers all the previously mentioned attacks? And if so, can it be achieved for FHE schemes that are based on bootstrapping techniques?

1.1 Our Contributions and Techniques

We answer these questions positively. We propose a new relaxation of CCA2, called *verified chosen-ciphertext attack* (vCCA). This notion is meant to verifiably capture the integrity of the ciphertext for the whole chain of FHE operations: from the encryption, the homomorphic evaluation to the decryption. The idea is that a malicious party that executes the evaluation algorithm, should be forced to use valid ciphertexts, and run the evaluation algorithm correctly. We show that our vCCA security notion strictly implies CCA1 security. Finally, we show that vCCA is achievable for FHE schemes by giving a general transformation that takes *any* CPA secure FHE scheme and makes it vCCA secure. This includes schemes based on bootstrapping.

Defining the vCCA notion. In the CCA2 game, an adversary that attacks an FHE scheme can always modify the challenge ciphertext using homomorphism before querying it to its decryption oracle. Our idea for vCCA is to capture schemes such that this is detectable under certain conditions. Essentially, we require that the ciphertext output by the evaluation algorithm is "linked" to the ciphertexts used as inputs. This allows us to define a modified vCCA oracle to characterize the vCCA security notion. This oracle, upon receiving a decryption query, checks whether it is linked to the challenge ciphertext, and performs decryption only if it is not the case.

We give two formulations of vCCA security. Our first definition is based on the indistinguishability game, where the oracle of the second phase is the vCCA oracle described above. Consequently, we call this notion *indistinguishability against verified chosen ciphertext-attack* (IND-vCCA). Our second definition uses the non-malleability simulation paradigm [30]. We use the vCCA oracle to generalize the targeted malleability notion from [14]. For every adversary that gets a tuple of ciphertexts and produces some output, there should exist a simulator that does not get these ciphertexts and produces an indistinguishable output. Our generalization, which we call TNM-vCCA, proceeds identically but replaces the oracles of the game with the vCCA oracle described above.

We show that TNM-vCCA is equivalent to IND-vCCA. This suggests that it is possible to think of TNM-vCCA as the non-malleability formulation of the IND-vCCA notion, similarly to the relationship between IND-CCA2 and NM-CCA2. It is interesting from a theoretical standpoint that the equivalence holds for this relaxed version of IND-CCA2 and NM-CCA2, whereas it is known that IND-CCA1 does not imply NM-CCA1 [7].

Relations between vCCA and other notions. To substantiate the validity of the definition of vCCA, we compare it with the existing notions that are achievable for FHE mentioned previously. We show that IND-vCCA implies them all. We also show that IND-vCCA is implied by RCCA (which is not achievable for FHE). We summarise our results in Figure 1. In the interest of readability, we do not include some of the known results between existing notions.

Constructing IND-vCCA-secure FHE. We show that IND-vCCA is achievable for FHE by giving a general framework to build IND-vCCA secure FHE

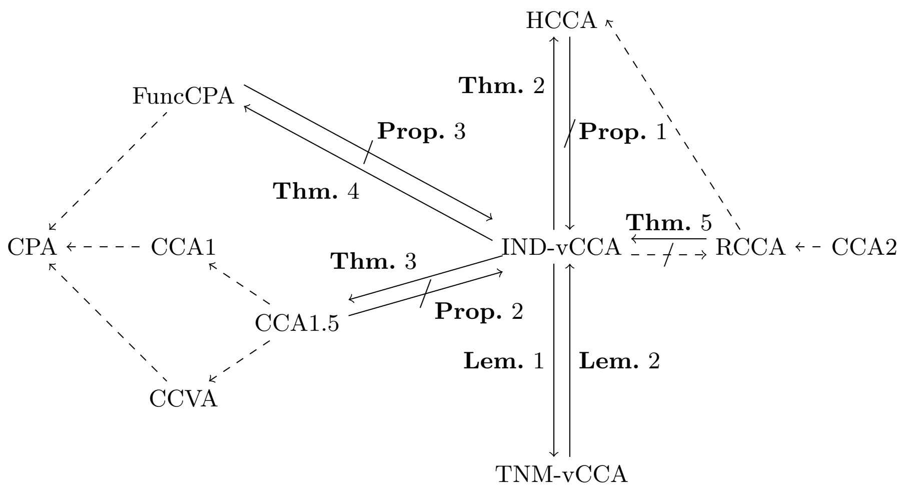

Fig. 1. Relation between the notions. If there is a path from A to B, then $A \implies B$. A hatched arrow represents a separation. A solid arrow is our result. A dashed arrow is a trivial result or a previous result.

schemes. The framework uses *any* CPA secure FHE scheme, and a succinct non-interactive argument of knowledge (SNARK). In particular, it supports schemes that are based on bootstrappping techniques. For instance, it can be built from state-of-the-art FHE schemes such as TFHE [25] or BGV [17]. The construction can also be used for approximate FHE schemes such as CKKS [24] with the caveat that the approximate FHE schemes need to be CPA^D secure. This stems from the fact that the common CCA2 transforms we use such as Encrypt-then-MAC or Naor-Yung double encryption assume the starting encryption scheme to be passively secure before preventing active attacks. In the case of approximate FHE, this has been shown to require CPA^D security [43]. We do not consider this further, and we will assume the FHE schemes to be exact in the rest of the paper.

Remark 1. In a recent paper, Checri et al. [22] have shown CPA^D attacks on non-approximate FHE schemes such as TFHE or BGV. However, to do so they "break" the correctness of the schemes, rendering them in essence "approximate". These attacks would carry over to our construction. Hence, it may be advisable in practice to implement the fixes they propose to get the CPA^D versions of these schemes as starting point for our construction. In this paper, we assume the homomorphic evaluations of the schemes to be correct, see Definition 3. As such, we only ask for CPA security (for exact FHE schemes). We discuss this further in Section 5.4.

Our framework proceeds along two separate steps. The first step is to make the FHE scheme CCA2 secure. This obviously implies that what is obtained after the first step is no longer a homomorphic encryption scheme. However, following ideas from Loftus et al. [44], we do it in such a way that part of the ciphertext keeps the homomorphic properties. We call such a procedure, *embedding the FHE scheme into a* CCA2 *encryption scheme.*

We propose four different ways to embed an FHE scheme into a CCA2 encryption scheme. We give embedding methods for public-key FHE schemes as well as symmetric-key FHE schemes. For each of these types, we consider two variants. In the first variant checking the validity of ciphertexts requires knowledge of the private key. We refer to such schemes as being *designated verifier*. In the second variant, checking the validity of ciphertexts does not require any secrets. We call such schemes *publicly verifiable*.

We use well-known paradigms to achieve CCA2 security for the embedding. In the symmetric case, we use Encrypt-then-MAC and Encrypt-then-Sign [8]. The latter one may be surprising, as Encrypt-then-Sign does not provide CCA2 security to PKE. However, here we are using it as part of a symmetric-key encryption scheme, essentially as a publicly verifiable message authentication code (MAC). In the asymmetric case, we use the double encryption paradigm by Naor and Yung [30,47]. In the publicly verifiable variant, we adopt Naor-Yung directly. However, in the designated-verfier case, we can remove the non-interactive zero-knowledge proof (NIZK) by using a CCA2 secure PKE. For the second ciphertext, we encrypt the concatenation of the actual message with the randomness used to do the first encryption. This implicitly acts as a NIZK and may be more efficient when using a PKE based for example on the Fujisaki-Okamoto transform [34].

The second step of the framework modifies the evaluation algorithm. We use a SNARK [12] to do the "linking" we require for IND-vCCA as mentioned above. More precisely, we make the evaluator prove that it did the homomorphic operation with valid input ciphertexts with respect to the CCA2 encryption scheme. Recall, the first step embeds the ciphertext of the FHE scheme in such a way that it is preserved after the embedding. This allows the evaluation algorithm to take that part as input, and then prove that it knows the corresponding "full" CCA2 ciphertext. This then allows to reduce any attack against the IND-vCCA security to the CCA2 security, modulo the soundness of the SNARK. We emphasize that since the bootstrapping keys are never valid ciphertexts for the CCA2 construction, an attacker cannot pass them through the evaluation algorithm as it will not be able to prove that it knows the corresponding valid ciphertext.

The designated-verifier constructions only achieve a relaxed version of compactness, i.e. the evaluated ciphertext grows linearly with the number of inputs to the evaluation algorithm as we actually must output back every input ciphertext in addition to the actual output of the homomorphic evaluation. This is necessary because the evaluator cannot check the validity of the ciphertexts when doing the SNARK proof. However, this also means that the designated-verifier construction has a more efficient evaluation algorithm than the publicly verifiable construction, exactly because the evaluator does not need to do the proof of validity. In contrast, the FHE scheme obtained with the publicly verifiable construction achieves standard compactness.

The standard security property regarding soundness for SNARKs is knowledge soundness. This asks that one should be able to extract a witness from a proof given white-box access to the prover. To achieve IND-vCCA security, we require somewhat stronger soundness properties from the SNARK. As IND-vCCA regulates the malleability of the FHE schemes, we need non-malleable proofs from the SNARK. This is caputred by a property called simulation-extractability. Furthermore, we require for our security reduction that the witness extraction can be done with only black-box access to the prover, i.e., without any knowledge about its code or the randomness it is using. Nonetheless, we also show that if our transformation is instantiated with a SNARK that is "just" (white-box) simulation-extractable, we still obtain a CCA1-secure FHE scheme.

1.2 Related Work

Earlier constructions of CCA1 secure FHE schemes did not support bootstrapping. Loftus et al. [44] gave a CCA1 secure SHE construction based on adding plaintext-awareness to a CPA secure SHE scheme, which is a known way to achieve CCA1 security [9]. However, the CPA security of their SHE was based on the short principal ideal problem (SPIP), which has later been shown to be insecure [10, 11, 27].

Canetti et al. [20] and Yasuda et al. [56] proposed concurrently similar CCA1 secure FHE schemes based on multi-key identity-based homomorphic encryption (IBHE) and multi-key homomorphic encryption respectively. Both constructions share the drawback that the evaluated ciphertext must contain all the identities used to do the encryption for the input ciphertexts. Therefore, evaluated ciphertexts grow linearly in the amount of inputs to the evaluation algorithm, resulting in non-compact FHE schemes.

A different CCA1 secure FHE construction, also proposed in [20], achieves compactness by using indistinguishability obfuscation (iO). Wang et al. [55] later identified and patched a security weakness in this scheme, but importantly it still hinges on iO. This is a major drawback to its practicality due to the heavy complexity price iO demands. Hence, this remains primarily a theoretical result pending some major new developments in the iO field. None of these schemes have been concretely implemented.

The integrity of FHE schemes as a byproduct of verifiability has been considered by Viand et al. [53, 54]. They define verifiable FHE (vFHE), which combines notions from verifiable computation - such as soundness - with confidentiality notions such as CCA1 security. Note, however, that the CCA1 definition used in [53, 54] is weaker than what would be expected from a standard CCA1 notion. In particular, the adversary in [53, 54] is limited in its use of the evaluation algorithm to a single function which is fixed at the beginning of the game. Moreover, the general vFHE construction in [53, 54] is a combination of a CPA secure FHE scheme with a SNARK. However, we highlight here that this construction is *not* CCA1 secure, neither wrt. the standard CCA1 definition nor wrt. the weaker definition used in [53, 54], as explained in the following. Their vFHE scheme uses a SNARK to prove that the evaluation algorithm has been run correctly. More precisely, let c_x be an encryption of some plaintext m. Then, the decryption algorithm only decrypts a tuple (c_y, π) if π is a valid proof that c_y is the output of the evaluation algorithm with f, c_x as inputs, where f is a function fixed at the key generation. However, this does not prevent any existing CCA1 attacks on the underlying FHE scheme. Any ciphertext that would be queried to a decryption oracle in the context of a CCA1 attack against the underlying FHE scheme can easily be converted into a ciphertext for the proposed vFHE scheme. Indeed, any queried ciphertext c can be put through the evaluation algorithm with the identity function as input, and compute the corresponding SNARK proof. This results in a valid ciphertext for the vFHE scheme, which can then be submitted to the decryption oracle.

The use of SNARKs in FHE constructions has been considered multiple times in the literature, either in the context of verifiable computations as mentioned previously, or to achieve higher levels of security. Boneh et al. [14] and Cannetti et al. [20] gave similar constructions based on the Naor-Yung paradigm, where the usual NIZK is replaced with a SNARK to achieve the required FHE compactness. An FHE ciphertext $c = (c_1, c_2, \pi)$ is a double encryption of a message with a proof π that c_1 and c_2 encrypt the same plaintext. The evaluation algorithm is performed on both c_1 and c_2 to maintain that invariant. This FHE approach is proven CCA1 secure in [20]. We note that one variant of our FHE constructions is closely related to this approach, as we also use the double encryption paradigm to embed the homomorphic encryption scheme. The main differences with our construction is that we extend this idea to support bootstrapping, and that we only require the execution of the evaluation algorithm once, instead of twice.

From a practical perspective, several works aimed at improving the efficiency of combining FHE with SNARKs. State-of-the-art FHE schemes work over lattices. This means that proving their evaluation algorithm amounts to proving ring operations. However, the most well-known SNARKs (e.g. [41]) work over groups. This means that proving the homomorphic evaluation requires to emulate the ring operations with group operations. To avoid this source of inefficiencies, Ganesh et al. [36] designed a SNARK that natively works over rings. However, the proof system in [36] is not suited for ciphertexts maintenance techniques such as bootstrapping or modulus switching. This has recently been addressed by Atapoor et al. [5]. Using lattice-based SNARKs, they were able to efficiently prove the modulus switching and key switching operations, in addition to the regular homomorphic operations.

Organisation of the paper. We give the notations and preliminaries in Section 2. In Section 3, we define the vCCA security notion, giving both the indistinguishability version as well as the non-malleability one. In Section 4, we start by establishing the equivalence of both formulations, before analysing the relations between vCCA and other security notions. In Section 5, we present the first step of our framework — embedding of an FHE scheme into a CCA2-secure encryption scheme and present four approaches for different types of schemes and ciphertext validity checks. In Section 6, we provide the second step of the framework — construction of an IND-vCCA-secure FHE scheme from a CCA2-secure encryption scheme with FHE embedding. We conclude in Section 7.

2 Preliminaries

Notations. Let $\mathcal{A}$ be a probabilistic algorithm. We write $c \leftarrow \mathcal{A}(x_1, \dots, x_l; r)$ to denote that $\mathcal{A}$ on input $x_1, \dots, x_l$ with randomness r outputs c. We may omit r and write $\mathcal{A}(x_1, \dots, x_l)$ to denote that we run the algorithm with a uniformly sampled r.

Definition 1 (PKE). *A public-key encryption scheme* $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *over a message space* $\mathcal{M}$ *is a triple of algorithms: a* PPT *key generation algorithm* Gen*, a* PPT *encryption algorithm* Enc *and a deterministic polynomial-time decryption algorithm* Dec *such that for any security parameter* λ*:*

1. $\mathsf{Gen}(1^\lambda)$*: Outputs a public key / secret key pair denoted* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. $\mathsf{Enc}_{\mathsf{pk}}(m)$*: On input a message* $m \in \mathcal{M}$ *and the public-key* pk*, output the encryption of* m *denoted* $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$.
3. $\mathsf{Dec}_{\mathsf{sk}}(c)$*: On input a ciphertext* c *and the secret key* sk*, output* $m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$ *with* $m \in \mathcal{M} \cup \{\perp\}$.

We require correctness. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *be a public key/ secret key pair. Then for all* $m \in \mathcal{M}$*,*

$$\Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m)) \neq m] \leq \mathsf{negl}(\lambda)$$

We call a ciphertext c *valid* if there exist $m \in \mathcal{M}$ and an appropriate value r such that $c = \mathsf{Enc}_{\mathsf{pk}}(m; r)$. Otherwise we say it is *invalid*. For a given encryption scheme $\mathcal{E}$, we say that a polynomial-time algorithm Vf is a verification algorithm for $\mathcal{E}$, if on input a ciphertext c it outputs whether or not it is valid. If Vf only requires public information, we say that $\mathcal{E}$ is *publicly verifiable*. If it requires the secret key, we say that $\mathcal{E}$ is *designated verifier*. In that case we define a canonical verification algorithm that runs $\mathsf{Dec}_{\mathsf{sk}}(c)$, and simply outputs "invalid" if Dec outputs $\perp$, and "valid" otherwise. As indicated, the keys are given as subscript to the algorithms. We may also explicitly write them as input to the algorithm if we want to emphasize which key is used when multiple exist in the setting.

Remark 2. For the rest of this paper, we take the convention from [29] and restrict the definition of PKE to require that the decryption algorithm outputs $\perp$ when it receives an invalid input. In the general case, this could also be a random value m in the message space.

We recall the usual security notions for encryption. For conciseness, we give a general indistinguishability game in Figure 2, that we specify with the string $atk \in \{\text{CPA}, \text{CCA1}, \text{CCA2}\}$ to determine which oracle should be considered.

```
Experiment $\mathrm{Exp}^{atk}_{\mathcal{E},\mathcal{A}}(\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
$(m_0, m_1, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_1(\cdot)}(1^\lambda, \mathsf{pk})$, where $|m_0| = |m_1|$
$b \xleftarrow{\$} \{0,1\}, c^* \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_b)$
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}_2(\cdot)}(\mathsf{pk}, s, c^*)$
return $b = b'$
```

Fig. 2. The indistinguishability experiment

Definition 2 (Indistinguishability [7]). *Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme. We say that $\mathcal{E}$ is indistinguishable against atk, if for any* PPT *adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage of $\mathcal{A}$ in the indistinguishability game defined in Figure 2 is negligible, where,*

$$\begin{array}{lll} \mathcal{O}_1(\cdot) = \epsilon & \mathcal{O}_2(\cdot) = \epsilon & \textit{for atk} = \mathrm{CPA} \\ \mathcal{O}_1(\cdot) = \mathsf{Dec}_{\mathsf{sk}}(\cdot) & \mathcal{O}_2(\cdot) = \epsilon & \textit{for atk} = \mathrm{CCA1} \\ \mathcal{O}_1(\cdot) = \mathsf{Dec}_{\mathsf{sk}}(\cdot) & \mathcal{O}_2(\cdot) = \mathsf{Dec}_{\mathsf{sk}}(\cdot) & \textit{for atk} = \mathrm{CCA2} \end{array}$$

with ϵ being the function which, on any input, returns the empty string. In the case of CCA2*, $\mathcal{A}_2$ is prohibited from asking its oracle to decrypt c^*. The advantage is defined as,*

$$\mathsf{Adv}^{\mathrm{atk}}_{\mathcal{E},\mathcal{A}}(\lambda) = |\Pr[\mathrm{Exp}^{atk}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$$

(Fully) Homomorphic Encryption. We recall here the definition for public-key (fully) homomorphic encryption schemes, adapted from [1]. In this paper, we also consider the symmetric-key version, but skip its definition in the interest of space.

Definition 3 ((Fully) Homomorphic Encryption). *Let $\mathcal{F}$ be a family of functions. A public-key $\mathcal{F}$-homomorphic encryption scheme $\mathsf{HE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ with message space $\mathcal{M}$ is a quadruple of* PPT *algorithms such that:*

- $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a correct* PKE
- Eval *takes as input the public key* pk*, a function $f \in \mathcal{F}$, and a tuple of ciphertexts $c_1, \ldots, c_l$, and outputs a ciphertext $\hat{c} \leftarrow Eval_{\mathsf{pk}}(f, c_1, \ldots, c_l)$.*
- Homomorphic correctness*: For all functions $f \in \mathcal{F}$ and for all plaintexts $m_1, \ldots, m_l \in \mathcal{M}$, if we let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_i)$ for $i \in \{1, \ldots l\}$, it holds that:*

$$\Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Eval}_{\mathsf{pk}}(f, c_1, \ldots, c_l)) \neq f(m_1, \ldots, m_l)] \leq \mathsf{negl}(\lambda)$$

We say that HE *is a fully homomorphic encryption (FHE) scheme if* $\mathcal{F}$ *is the set of all the efficiently computable functions.*

Typically, fully homomorphic encryption schemes require a property called *compactness*. This asks that the size of the evaluated ciphertexts does not grow trivially with the size of the evaluated function. We will also consider *relaxed compactness* where we allow the evaluated ciphertext to grow linearly with the number of inputs to the evaluation algorithm.

Some FHE schemes output a *bootstrapping key* bk. We implicitly include it as part of the public key pk. In some cases, we may explicitly write it out when necessary for clarity. In that case, $\mathsf{Gen}(1^\lambda)$ generates $(\mathsf{pk}, \mathsf{sk}, \mathsf{bk})$ and the Eval algorithm takes bk as input instead of pk. We call a ciphertext c a *fresh ciphertext* if it is the output of the encryption algorithm. If it is the output of the evaluation algorithm, we call it an *evaluated ciphertext*.

MAC and Digital Signature. We use the standard definitions for message authentication codes (MAC) $\mathcal{I} = (\mathcal{I}.\mathsf{Gen}, \mathcal{I}.\mathsf{Tag}, \mathcal{I}.\mathsf{Vfy})$ and digital signatures $\mathcal{S} = (\mathcal{S}.\mathsf{Gen}, \mathcal{S}.\mathsf{Sign}, \mathcal{S}.\mathsf{Vfy})$. We require for both strong unforgeability against chosen-message attacks (SUF-CMA). See, e.g., [8, 15] for corresponding definitions.

SNARK In our transformation we use a succinct non-interactive argument of knowledge (SNARK). Apart from its usual properties of completeness and succinctness, we ask for two somewhat more uncommon properties. First we require non-malleability, which is modelled as simulation-sound extractability (SE). This asks that an adversary should not be able to "maul" a proof into another valid proof. More precisely, there must exists an extraction algorithm that can recover a witness for the proven statement from any valid proof, even if the adversary has access to a simulated proof oracle. SNARKs usually achieve white-box extractability where the extraction algorithm may depend on the adversary. We further require black-box extraction where there exists a single witness extractor that works for all adversaries.The following definitions are inspired from [6, 41].

Let $\mathcal{R}_\lambda$ be an efficiently computable binary relation which consists of pairs of the form (x, w), where x is a statement, and w is a witness. Let L be the language associated with the relation $\mathcal{R}_\lambda$, i.e. $L = \{x \mid \exists w\ \mathcal{R}(x, w) = 1\}$. A non-interactive argument for an NP language L is a quadruple of PPT algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim})$ such that

– $(\sigma, \tau, \tau_{\text{ext}}) \leftarrow \mathsf{Setup}(\mathcal{R}_\lambda)$: The setup porduces a common reference string σ and trapdoors τ, τ_{ext} for the relation $\mathcal{R}_\lambda$.
– $\pi \leftarrow \mathsf{Prove}(\sigma, x, w)$: Takes as input σ, a statement witness pair $(x, w) \in \mathcal{R}_\lambda$ and produces an argument π.
– $b \leftarrow \mathsf{Verify}(\sigma, x, \pi)$: Takes as input a σ, a statement x and a proof π and outputs a bit b denoting accept or reject.
– $\pi \leftarrow \mathsf{Sim}(\tau, x)$: Takes as input a simulation trapdoor τ and a statement x and outputs an argument π

Definition 4. *We say that $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim})$ is a succinct non-interactive argument of knowledge (*SNARK*) if it is a non-interactive argument with perfect completeness, black-box weak simulation-sound extractability and succinctness as defined below.*

Perfect completeness: For every true statement for the relation $\mathcal{R}_\lambda$, an honest prover with a valid witness always convinces the verifier: $\forall (x, w) \in \mathcal{R}_\lambda$:

$$\Pr[\mathsf{Verify}(\sigma, x, \pi) = 1 \mid (\sigma, \tau, \tau_{\text{ext}}) \leftarrow \mathsf{Setup}(\mathcal{R}_\lambda); \pi \leftarrow \mathsf{Prove}(\sigma, x, w)] = 1$$

Zero-knowledge: We say that Π is zero-knowledge if for all PPT adversaries and all $(x, w) \in \mathcal{R}_\lambda$:

$$\begin{aligned} &\Pr[\mathcal{A}(\sigma, \tau, \pi) = 1 \mid (\sigma, \tau, \tau_{\text{ext}}) \leftarrow \mathsf{Setup}(\mathcal{R}_\lambda); \pi \leftarrow \mathsf{Prove}(\sigma, x, w)] \\ -&\Pr[\mathcal{A}(\sigma, \tau, \pi) = 1 \mid (\sigma, \tau, \tau_{\text{ext}}) \leftarrow \mathsf{Setup}(\mathcal{R}_\lambda); \pi \leftarrow \mathsf{Sim}(\tau, x)] \leq \mathsf{negl}(\lambda) \end{aligned}$$

Black-box weak SE: We say that Π is black-box weak SE if there exists a PPT witness extraction algorithm Ext, such that for any PPT adversary $\mathcal{A}$ and $\mathcal{R}_\lambda$,

$$\Pr\left[\begin{array}{c|c} \mathsf{Verify}(\sigma, x, \pi) = 1 \wedge & (\sigma, \tau, \tau_{\text{ext}}) \leftarrow \mathsf{Setup}(\mathcal{R}_\lambda) \\ (x, w) \notin \mathcal{R}_\lambda \wedge x \notin \mathcal{Q} & (x, \pi) \leftarrow \mathcal{A}^{S_{\sigma,\tau}}(\sigma); w \leftarrow \mathsf{Ext}(\sigma, \tau_{\text{ext}}, x, \pi) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where $S_{\sigma,\tau}(x)$ is a simulator oracle that calls $\mathsf{Sim}(\sigma, \tau, x)$ internally, and also records x into $\mathcal{Q}$.
Succinctnesss: The verifier runs in polynomial time in $\lambda + |x|$ and the proof size is polynomial in λ.

Ganesh et al. [35] gave a construction for a universal composable (UC) SNARK in the global random oracle model (GROM). It is known that a UC secure SNARK must be black-box SE [40], and as such is suitable for our construction.

3 Verified CCA (vCCA) Security

In this section, we formally introduce vCCA security, and give two equivalent formulations of it. Our security definition aims to capture the intuitive notion of controlling the malleability of the ciphertexts to ensure some form of integrity.

In the CCA2 game, the adversary is prohibited to make a decryption query for the challenge ciphertext. This is the minimal restriction that has to be placed for the notion to be achievable. Previous relaxations of CCA2, such as RCCA [19] or generalized CCA (gCCA) [3] have relaxed the non-malleability property by increasing this restriction. The "illegal decryption query" set is expanded to contain ciphertexts that are related to the challenge ciphertext; in gCCA, the set is every ciphertext that satisfy a certain binary relation with the challenge ciphertext while in RCCA, the illegal set is the set of every ciphertext such that their decryption is one of the challenge plaintexts.

As noted in [49], generalizing this approach further for homomorphic encryption quickly expands the illegal set to be the whole ciphertext space, making the notion equivalent to CCA1; this holds even for schemes that only support homomorphism over univariate functions.

Our solution is to require the evaluated ciphertext to be linked to the input ciphertext. Our illegal set is then every ciphertext which is linked to the challenge ciphertext through the evaluation algorithm. In more detail, we will require the existence of an extraction algorithm that on input a ciphertext, will recover a statement describing if the ciphertext is the output of the evaluation algorithm. If so, the statement must contain the function f and the ciphertexts $\{c_1, \dots c_l\}$ that were the input to the evaluation algorithm. If the ciphertext c is a fresh output from the encryption algorithm, the extraction algorithm returns a special symbol e.g. $\perp$ or (Id, c) where Id is the identity function.

Definition 5 (Extraction algorithm). *Let* $\mathsf{HE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *be a* $\mathcal{F}$*-homomorphic encryption scheme with ciphertext-space* $\mathcal{C}$ *and let* $\mathcal{F}'$ *be a subset of* $\mathcal{F}$*. A* $\mathcal{F}'$*-extraction algorithm* $\mathsf{Extract}$ *for* HE *is a* PPT *algorithm that takes as input a ciphertext* $c \in \mathcal{C}$*, optionally some auxiliary information* aux*, and outputs a statement* $\Phi \in \mathcal{F}' \times \mathcal{C}^l \cup \{\perp\}$*.*

Let $\mathcal{A}$ *be a* PPT *adversary and let* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *be a key-pair. Let* c *be a ciphertext output by* $\mathcal{A}(\mathsf{pk})$*, and let* $(f, c_1, \dots, c_l)$ *be the output of* $\mathsf{Extract}(\mathrm{aux}, c)$*. We say that* $\mathsf{Extract}$ *is sound if every* $c_i \in \{c_1, \dots, c_l\}$ *is a fresh ciphertext and the following two distributions are indistinguishable, i.e.*

$$\{\mathsf{Dec}_{\mathsf{sk}}(c)\} \approx \{f(\mathsf{Dec}_{\mathsf{sk}}(c_1), \dots, \mathsf{Dec}_{\mathsf{sk}}(c_l))\}$$

We consider that $f(\mathsf{Dec}_{\mathsf{sk}}(c_1), \dots, \mathsf{Dec}_{\mathsf{sk}}(c_l)) = \perp$ if there exists an i such that $\mathsf{Dec}_{\mathsf{sk}}(c_i) = \perp$. The auxiliary input may include the public-key of the encryption scheme, but should not contain the secret-key. If $\mathcal{F} = \mathcal{F}'$, we may just write that $\mathsf{Extract}$ is an extraction algorithm for HE. As an abuse of notation, we may write $c \in (f, c_1, \dots, c_l)$ to denote $c \in \{c_1, \dots, c_l\}$. We now give the main definition for IND-vCCA.

We use the extraction algorithm to define the vCCA oracle in Figure 3. We then use it to define the IND-vCCA security notion. It is exactly the IND-CCA2 game, with the oracle for the second phase replaced by the vCCA oracle.

Definition 6 (IND-vCCA **Security).** *Let* $\mathsf{HE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *be a* $\mathcal{F}$*-homomorphic encryption scheme. We say that* HE *is indistinguishable against verified adaptive chosen-ciphertext attacks (*IND-vCCA*) secure with respect to* $\mathcal{F}'$*, if there exists an* $\mathcal{F}'$*-extraction algorithm* $\mathsf{Extract}$ *for* HE *such that for any* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, the advantage of* $\mathcal{A}$ *in the following experiment* $\mathrm{Exp}^{\text{IND-vCCA}}_{\mathsf{HE},\mathcal{A}}(\lambda)$ *is negligible.*

```
Oracle $\mathcal{O}_{\text{vCCA}}(c^*, c)$
$(f, c_1, \ldots c_l) \leftarrow \mathsf{Extract}(\text{aux}, c)$
if $c^* \in \{c_1, \ldots c_l\}$
  return $\perp$
else
  return $\mathsf{Dec}_{\mathsf{sk}}(c)$
```

Fig. 3. The vCCA oracle

```
Experiment $\text{Exp}^{\text{IND-vCCA}}_{\mathcal{E},\mathcal{A}}(\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
$(m_0, m_1, s) \leftarrow \mathcal{A}_1^{\mathsf{Dec}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$
$b \xleftarrow{\$} \{0, 1\}$
$c^* \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_b)$
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{vCCA}}(c^*, \cdot)}(\mathsf{pk}, s, c^*)$
return $(b = b')$
```

where $|m_0| = |m_1|$*. The advantage of* $\mathcal{A}$ *is defined as:*

$$\mathsf{Adv}^{\text{IND-vCCA}}_{\mathcal{E},\mathcal{A}}(\lambda) = |\Pr[\text{Exp}^{\text{IND-vCCA}}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$$

Targeted Malleability. Boneh et al. [14] defined targeted malleability in order to "restrict" the malleability of an FHE scheme. This enforces some form of control on the evaluation algorithm, and makes it a close notion to IND-vCCA. The original definitions of targeted malleability are in the context of CPA and CCA1 attacks. We extend the notion in Definition 7 by adding our vCCA oracle as we did for IND-vCCA.

Definition 7 (TNM-vCCA)**.** *Let* $t = t(\lambda)$ *be a polynomial. A public-key encryption scheme* $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is t-bounded non-malleable against verified chosen-ciphertext attacks with respect to a set of functions* $\mathcal{F}$ ($\mathcal{F}$-TNM-vCCA) *if for any polynomials* $r = r(\lambda)$ *and* $q = q(\lambda)$ *and for any probabilistic polynomial-time algorithm* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a probabilistic polynomial-time algorithm* $S = (S_1, S_2)$ *such that the following distributions* $\{\text{Real}^{\text{TNM-vCCA}}_{\mathcal{E},\mathcal{A},t,r,q}(\lambda)\}_{\lambda\in\mathbb{N}}$ *and* $\{\text{Sim}^{\text{TNM-vCCA}}_{\mathcal{E},S,t,r,q}(\lambda)\}_{\lambda\in\mathbb{N}}$ *(see Figure 4) are computationally indistinguishable.*

We write TNM-vCCA when the set $\mathcal{F}$ is clear from context. It is clear that this generalization implies the CCA1 and CPA versions defined in [14]. Furthermore, it should be clear from Definition 6 that IND-vCCA implies CCA1.

```
$\mathrm{Real}^{\text{TNM-vCCA}}_{\mathcal{E},\mathcal{A},t,r,q}(\lambda)$
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
$(\mathcal{M}, s_1, s_2) \leftarrow \mathcal{A}_1^{\mathsf{Dec}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$
$(m_1, \ldots, m_r) \leftarrow \mathcal{M}$
$c_i^* \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_i)$ for $i \in \{1, \ldots, r\}$
$(c_1, \ldots c_q) \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{vCCA}}(c^*,\cdot)}(c_1^*, \ldots, c_r^*, s_2)$
for $j \in \{1, \ldots, q\}$ do
    if $c_j = c_i^*$ do
        $d_j \leftarrow \mathsf{copy}_i$
    else do
        $d_j \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c_j)$
return $(s_1, m_1, \ldots, m_r, d_1, \ldots, d_q)$
```

```
$\mathrm{Sim}^{\text{TNM-vCCA}}_{\mathcal{E},S,t,r,q}(\lambda)$
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
$(\mathcal{M}, s_1, s_2) \leftarrow S_1(\mathsf{pk})$
$(m_1, \ldots, m_r) \leftarrow \mathcal{M}$
$(c_1, \ldots c_q) \leftarrow S_2(s_2)$
for $j \in \{1, \ldots, q\}$ do
    if $c_j = \mathsf{copy}_i$ do
        $d_j \leftarrow \mathsf{copy}_i$
    if $c_j = (i, f_1, \ldots, f_l)$,
        with $i \in \{1, \ldots, r\}$
        $l \leq t \wedge f_1, \ldots, f_l \in \mathcal{F}$ do
        $d_j \leftarrow f(m_i)$ with $f = f_1 \circ \cdots \circ f_l$
    else do
        $d_j \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c_j)$
return $(s_1, m_1, \ldots, m_r, d_1, \ldots, d_q)$
```

Fig. 4. The distribution $\{\mathrm{Real}^{\text{TNM-vCCA}}_{\mathcal{E},\mathcal{A},t,r,q}(\lambda)\}_{\lambda\in\mathbb{N}}$ and $\{\mathrm{Sim}^{\text{TNM-vCCA}}_{\mathcal{E},S,t,r,q}(\lambda)\}_{\lambda\in\mathbb{N}}$

4 Relations between vCCA Security and Other Notions

We start by showing that our formulations of vCCA security, namely IND-vCCA and TNM-vCCA, are equivalent. We then show that IND-vCCA is the strongest notion among the notions we consider that are achievable for FHE.

4.1 IND-vCCA and TNM-vCCA Are Equivalent

As mentioned, being able to restrict a *fully* homomorphic encryption scheme to a scheme that only allows homomorphism for a smaller subset $\mathcal{F}$ implies some form of control on the evaluation algorithm. In fact, we show that for any $\mathcal{F}$-homomorphic encryption scheme, IND-vCCA security with respect to $\mathcal{F}'$ is equivalent to $\mathcal{F}'$-TNM-vCCA.

Theorem 1. *Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a $\mathcal{F}$-homomorphic encryption scheme and let $\mathcal{F}'$ be a subset of $\mathcal{F}$. Then $\mathcal{E}$ is* IND-vCCA *secure with respect to $\mathcal{F}'$ if and only if it is* TNM-vCCA *secure with respect to $\mathcal{F}'$.*

We give the proof for the case that $r = q = 1$. The proof generalizes naturally to the other cases. We separate the proof into two lemmas. We start by showing that IND-vCCA $\Longrightarrow$ TNM-vCCA. The proof follows closely ideas given in the proof of TNM-vCCA-security given in [14, Section 4.4], and can be seen as a generalization of it. The main difference is that we use the extraction algorithm given by the IND-vCCA property to create the "certification chain" that they use.

Lemma 1. IND-vCCA $\implies$ TNM-vCCA

Proof. Assume that $\mathcal{E}$ is IND-vCCA secure with respect to a set $\mathcal{F}'$ and let $\mathsf{Extract}$ be the associated extraction algorithm. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against the TNM-vCCA security of $\mathcal{E}$. We build a simulator $S = (S_1, S_2)$ that uses $\mathcal{A}$ as a subroutine.

On input pk, S_1 generates its own key-pair $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Gen}(1^\lambda)$. It then runs $(\mathcal{M}, s_1, s_2) \leftarrow \mathcal{A}_1(\mathsf{pk}')$. Any decryption queries are answered using sk'. S_1 then finally outputs $(\mathcal{M}, s_1, s_2')$ where $s_2' = (s_2, \mathcal{M}, \mathsf{pk}, \mathsf{pk}', \mathsf{sk}')$.

On input $s_2' = (s_2, \mathcal{M}, \mathsf{pk}, \mathsf{pk}', \mathsf{sk}')$, S_2 draws $m' \leftarrow \mathcal{M}$ and computes $c^* \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m')$. It then runs $c \leftarrow \mathcal{A}_2(c^*, s_2)$. Any decryption queries to the vCCA oracle are answered by using sk' and the extraction algorithm. S_2 then uses the extraction algorithm on c to obtain $\Phi = (f, c_1 \dots c_l)$. If c is not valid in the sense of TNM-vCCA, i.e. $\mathsf{Dec}_{\mathsf{sk}'}(c) = \perp$, then it outputs $\perp$. Otherwise:

- If $c = c^*$, S_2 outputs copy_1.
- If $c^* \in \Phi$, S_2 uses sk' to obtain $m_i \leftarrow \mathsf{Dec}_{\mathsf{sk}'}(c_i)$ for every $c_i \in \Phi$. It then outputs $(f, m_1, \dots, 1, \dots, m_l)$ (the 1 indicates where the challenge plaintext is placed).
- If $c^* \notin \Phi$, S_2 first computes $c_i' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mathsf{Dec}_{\mathsf{sk}'}(c_i))$ for every $c_i \in \Phi$. It then outputs $c' \leftarrow \mathsf{Eval}_{\mathsf{pk}}(f, c_1', \dots c_l')$.

We define distributions $\mathcal{D}_1, \dots \mathcal{D}_3$ where $\mathcal{D}_1 = \mathrm{Sim}^{\text{TNM-vCCA}}_{\mathcal{E},S,t,r,q}(\lambda)$ and $\mathcal{D}_3 = \mathrm{Real}^{\text{TNM-vCCA}}_{\mathcal{E},\mathcal{A},t,r,q}(\lambda)$.

Distribution $\mathcal{D}_1$. This is the distribution given by $\mathrm{Sim}^{\text{TNM-vCCA}}_{\mathcal{E},S,t,r,q}(\lambda)$.

Distribution $\mathcal{D}_2$. This distribution is the same as $\mathcal{D}_1$ apart from the following modification. c^* is now computed from m instead of m', i.e. $c^* \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$.

Distribution $\mathcal{D}_3$. This is the distribution given by $\mathrm{Real}^{\text{TNM-vCCA}}_{\mathcal{E},\mathcal{A},t,r,q}(\lambda)$.

The rest of the proof follows [14, Section 4.4], with the following modification. We use the IND-vCCA property to go from $\mathcal{D}_1$ to $\mathcal{D}_2$. We use the soundness of the extraction algorithm to go from $\mathcal{D}_2$ to $\mathcal{D}_3$. We sketch out the rest of the proof for completeness.

The indistinguishably of distribution $\mathcal{D}_1$ and $\mathcal{D}_2$ follows from the IND-vCCA property. Indeed, S does not have access to any oracle of its own, and perfectly simulates the vCCA oracle for $\mathcal{A}$.

In both $\mathcal{D}_2$ and $\mathcal{D}_3$, the algorithm $\mathcal{A}_2$ is given $c^* \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$ as input and outputs a ciphertext c. Notice that in both distribution, if c is invalid the output is $(s_1, m, \perp)$ and if $c = c^*$, the output is $(s_1, m, \mathsf{copy}_1)$.

When c is valid and $c \neq c^*$, the output in $\mathcal{D}_3$ is $(s_1, m, \mathsf{Dec}_{\mathsf{sk}}(c))$. We show that this is also the case in $\mathcal{D}_2$ with overwhelming probability. The soundness property of the extraction algorithm implies that the probability that $c \neq \mathsf{Eval}(f, c_1, \dots, c_l)$ is negligible. Assume in the following that $c = \mathsf{Eval}(f, c_1, \dots, c_l)$. Consider the two possible cases:

- If $c \in \Phi$, the output of $\mathcal{D}_2$ is $(s_1, m, f(m_1, \ldots, m_l))$ with $m = m_j$ for some j. Since $c = \mathsf{Eval}(f, c_1, \ldots, c_l)$, it follows from the correctness of $\mathcal{E}$ that with overwhelming probability $\mathsf{Dec}_{\mathsf{sk}}(c) = f(m_1, \ldots, m_l)$.
- If $c \notin \Phi$, the output of $\mathcal{D}_2$ is $(s_1, m, \mathsf{Dec}_{\mathsf{sk}}(\mathsf{Eval}(f, c'_1, \ldots c'_l)))$ where every c'_i is a "translation" of c_i under pk. And the homomorphic correctness of the scheme implies that $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Eval}(f, c'_1, \ldots c'_l)) = f(\mathsf{Dec}_{\mathsf{sk}}(c'_1), \ldots, \mathsf{Dec}_{\mathsf{sk}}(c'_l))$. It follows from the correctness of the scheme that $\mathsf{Dec}_{\mathsf{sk}'}(c_i) = \mathsf{Dec}_{\mathsf{sk}}(c'_i)$, so $f(\mathsf{Dec}_{\mathsf{sk}}(c'_1), \ldots, \mathsf{Dec}_{\mathsf{sk}}(c'_l)) = f(\mathsf{Dec}_{\mathsf{sk}'}(c_1), \ldots, \mathsf{Dec}_{\mathsf{sk}'}(c_l)) = f(m_1, \ldots, m_l)$. Finally, as in the other case $\mathsf{Dec}_{\mathsf{sk}}(c) = f(m_1, \ldots, m_l)$. It follows that the output of $\mathcal{D}_2$ is $(s_1, m, \mathsf{Dec}_{\mathsf{sk}}(c))$ with overwhelming probability.

Hence, distribution $\mathcal{D}_1$ is indistinguishable from $\mathcal{D}_3$ and the lemma follows. □

Lemma 2. TNM-vCCA $\implies$ IND-vCCA.

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an IND-vCCA adversary against $\mathcal{E}$. We build an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ for the TNM-vCCA property.

Upon receiving the public-key pk, $\mathcal{B}_1$ runs $(m_0, m_1, s) \leftarrow \mathcal{A}_1(\mathsf{pk})$. It answers any decryption queries by using its own decryption oracle. $\mathcal{B}_1$ then outputs $(\mathcal{M}, s_1, s_2)$ where $\mathcal{M}$ is the uniform distribution over $\{m_0, m_1\}$, $s_1 = (m_0, m_1)$ and $s_2 = (m_0, m_1, s, \mathsf{pk})$.

On input (c^*, s_2), where $s_2 = (m_0, m_1, s, \mathsf{pk})$, $\mathcal{B}_2$ runs $b \leftarrow \mathcal{A}_2(\mathsf{pk}, c^*, s)$. Again, $\mathcal{B}_2$ answers every decryption query by using its own decryption oracle. Finally, it outputs $\mathsf{Enc}_{\mathsf{pk}}(f(m_b))$ for some $f \in \mathcal{F} \setminus \mathcal{F}'$. $\mathcal{B}$ perfectly simulates the IND-vCCA game for $\mathcal{A}$ and the lemma follows. □

4.2 vCCA Implies HCCA

Prabhakaran and Rosulek [49] introduced Homomorphic-CCA (HCCA) security as a generalization of PKE security notions such as RCCA [19] and gCCA [3]. HCCA captures chosen-ciphertext security for homomorphic encryption schemes but with significant limitation that it applies only to univariate functions. Our IND-vCCA can be seen as the generalization of HCCA to *fully* homomorphic encryption schemes.

Definition 8 (Homomorphic-CCA [49]). *Let $\mathcal{T}$ be a subset of the univariate functions. A homomorphic encryption scheme is Homomorphic-CCA (*HCCA*) secure with respect to $\mathcal{T}$ if there are* PPT *algorithms* RigEnc *and* RigExtract*, where the range of* RigExtract *is $\mathcal{T} \cup \{\perp\}$, and such that for all* PPT *adversaries $\mathcal{A}$,*

1. *The oracle* RigEnc *has an output of the form $(\zeta, S) \leftarrow \mathsf{RigEnc}_{\mathsf{pk}}$, such that the ζ is indistinguishable from an output from the encryption oracle* Enc.
2. *The advantage of $\mathcal{A}$ in the following* HCCA *experiment is negligible:*
   (a) *Setup: Pick $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and give* pk *to $\mathcal{A}$.*
   (b) *First Phase: $\mathcal{A}$ gets access to the $\mathsf{Dec}_{\mathsf{sk}}(\cdot)$ oracle and the following two "guarded"* RigEnc *and* RigExtract *oracles:*

  - $\mathsf{GRigEnc}_{\mathsf{pk}}() = \zeta_i$ *where* $(\zeta_i, S_i) \leftarrow \mathsf{RigEnc}_{\mathsf{pk}}$, *when called for the ith time*
  - $\mathsf{GRigExtract}_{\mathsf{sk}}(\zeta, i) = \mathsf{RigExtract}_{\mathsf{sk}}(\zeta, S_i)$

(c) *Challenge: $\mathcal{A}$ outputs a plaintext m^*. We privately flip a coin $b \leftarrow \{0,1\}$. If $b = 0$, we compute $\zeta^* \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m^*)$. If $b = 1$, we compute $(\zeta^*, S^*) \leftarrow \mathsf{RigEnc}_{\mathsf{pk}}$. In both cases, we give ζ^* to $\mathcal{A}$.*

(d) *Second Phase: $\mathcal{A}$ gets access to the same* GRigEnc *and* GRigExtract *oracles as in the First Phase, as well as a "rigged" version of the decryption oracle* RigDec. *When $b = 0$,* RigDec *is simply the normal decryption oracle $\mathsf{Dec}_{\mathsf{sk}}(\cdot)$. When $b = 1$,* RigDec *is implemented as follows:*

$$\mathsf{RigDec}_{\mathsf{sk}}(\zeta) = \begin{cases} T(m^*), & \textit{if } \perp \neq T \leftarrow \mathsf{RigExtract}_{\mathsf{sk}}(\zeta, S^*) \\ \mathsf{Dec}_{\mathsf{sk}}(\zeta), & \textit{otherwise} \end{cases}$$

(e) *Output: $\mathcal{A}$ outputs a bit b'. The advantage of $\mathcal{A}$ in this experiment is* $\mathsf{Adv}^{\mathrm{HCCA}}_{\mathsf{HE},\mathcal{A}}(\lambda) = |\Pr[b' = b] - \frac{1}{2}|$.

We show that IND-vCCA with respect to a set of univariate functions closed under composition strictly implies HCCA.

Theorem 2. *Let $\mathsf{HE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a $\mathcal{F}$-homomorphic encryption scheme and let $\mathcal{F}'$ be a subset of $\mathcal{F}$ that contains only univariate function and is closed under composition. Then* IND-vCCA*-security with respect to $\mathcal{F}'$ implies* HCCA*-security with respect to $\mathcal{F}'$. More precisely, for every* PPT *adversary $\mathcal{A}$, there exists a* PPT *adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathrm{HCCA}}_{\mathsf{HE},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{IND\text{-}vCCA}}_{\mathsf{HE},\mathcal{B}}(\lambda)$$

Proof. We now show that IND-vCCA implies HCCA. Assume HE is IND-vCCA and let Extract be the associated $\mathcal{F}'$-extraction algorithm. Let $\mathcal{A}$ be an HCCA adversary against HE. We build a IND-vCCA adversary $\mathcal{B}$ that uses $\mathcal{A}$. Upon receiving the public-key pk, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pk})$ and answers queries to the oracles in the following way:

- $\mathsf{GRigEnc}()$: Draw a random plaintext r_i and compute $(\zeta_i = \mathsf{Enc}_{\mathsf{pk}}(r_i), r_i)$. Return ζ_i.
- $\mathsf{GRigExtract}(\zeta, i)$: Compute $(f, c) \leftarrow \mathsf{Extract}(\mathrm{aux}, \zeta)$ where aux is the relevant auxialiary input. If $c = \zeta_i$, return f, otherwise return $\perp$.
- Challenge encryption: When $\mathcal{A}$ sends the challenge plaintext m^*, $\mathcal{B}$ first draws a random plaintext m' and sends (m^*, m') to its challenger. $\mathcal{B}$ receives an answer c^* and gives it to $\mathcal{A}$.
- $\mathsf{RigDec}(\zeta)$: Compute $(f, c) \leftarrow \mathsf{Extract}(\mathrm{aux}, \zeta)$. If $c = c^*$, return $f(m^*)$. Else $\mathcal{B}$ queries ζ to its decryption oracle and returns the answer to $\mathcal{A}$.

When $\mathcal{A}$ halts and outputs a guess $\hat{b}$, $\mathcal{B}$ also outputs $\hat{b}$. The soundness of the extraction algorithm implies that $\mathcal{B}$ perfectly simulates the HCCA game for $\mathcal{A}$ and the theorem follows. □

We now separate HCCA from IND-vCCA by showing that the HCCA construction given in [49, Section 5] is not IND-vCCA secure. Indeed they have shown this construction to also be unlinkable [49, Definition 3], and we show that this is incompatible with IND-vCCA security. The proof of Proposition 1 is given in Appendix A.

Proposition 1. HCCA $\not\Longrightarrow$ IND-vCCA

4.3 vCCA and Chosen-Ciphertext Verification Attacks

The chosen-ciphertext verification attacks (CCVA) notion is a variation of the chosen-ciphertext attacks notion, where the adversary gets access to a verification oracle instead of a decryption oracle. This was first considered in [42] under the name illegal ciphertext attacks (IND-ICA), and covers very practical attacks such as Bleichenbacher's attack on RSA-PKCS #1 [13]. More importantly for our purposes, Loftus et al. [44] have shown that their CCA1 secure FHE scheme, suffers from a CCVA attack. This underlines the relative strength of CCVA attacks.

We consider the version of the notion introduced by Das et al. [29] called CCA1.5. It is an indistinguishabillity game that combines the decryption oracle of the CCA1 game with the verification oracle from the CCVA game. This results in a stronger notion that implies both. We show here that IND-vCCA strictly implies CCA1.5. The following definition is adapted from [29, Definition 2].

Definition 9 (CCA1.5 **[29]**)**.** *An encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be* CCA1.5 *secure, if any* PPT *adversary $\mathcal{A}$ has negligible advantage in the bit-guessing version of the indistinguishability game defined in Figure 2, with the following oracles.*

- $\mathcal{O}_1(c)$*: Return* $\mathsf{Dec}_{\mathsf{sk}}(c)$
- $\mathcal{O}_2(c)$*: If c is a valid ciphertext, return "valid". Else, return "invalid".*

Theorem 3. IND-vCCA $\Longrightarrow$ CCA1.5

Proof. Note that an IND-vCCA adversary $\mathcal{B}$ that simulates the CCA1.5 game has no difficulty answering decryption queries in the first phase as it has its own decryption oracle. In the second phase, $\mathcal{B}$ must use its decryption oracle as a verification oracle. In the case that a verification query comes in for a ciphertext derived from the challenge ciphertext, the decryption oracle will return $\perp$, even for valid ciphertexts. Therefore, when $\mathcal{B}$'s decryption oracle returns $\perp$, it must check if the decryption failed because the ciphertext is related to the challenge ciphertext by using the extraction algorithm.

Let $\mathsf{HE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a $\mathcal{F}$-homomorphic encryption scheme. Assume that it is IND-vCCA secure and let $\mathsf{Extract}$ be the associated extraction algorithm. Let $\mathcal{A}$ be a PPT adversary against the CCA1.5 security of HE, we use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the IND-vCCA security of HE. On input the public-key pk, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pk})$ and answers queries in the following way.

- **The challenge query:** (m_0, m_1) is sent to $\mathcal{B}$'s challenger and the answer c^* is given to $\mathcal{A}$.
- **Decryption queries:** $\mathcal{B}$ simply uses its own decryption oracle.
- **Verification queries**: On input c, if $c = c^*$, $\mathcal{B}$ returns "valid". Otherwise, $\mathcal{B}$ queries its decryption oracle on c and receives an answer m. If $m \neq \perp$, $\mathcal{B}$ returns "valid". Else $\mathcal{B}$ uses the extraction algorithm to obtain $(f, c_1, \ldots, c_l) \leftarrow \mathsf{Extract}(c)$. Then, $\mathcal{B}$ verifies every c_i by querying every ciphertext c_i to its decryption oracle and recording the answer m_i in a list L. If one of the ciphertext c_j is such that $c_j = c^*$, it records m_j for some random $m_j \neq \perp$. Finally, if for every $m_i \in L$, $m_i \neq \perp$, it outputs "valid". Else it outputs "invalid".

Recall that the soundness of the extraction algorithm implies that if we have $(f, c_1, \ldots, c_l) \leftarrow \mathsf{Extract}(c)$, then $\{\mathsf{Dec}_{\mathsf{sk}}(c)\} \approx \{f(\mathsf{Dec}_{\mathsf{sk}}(c_1), \ldots, \mathsf{Dec}_{\mathsf{sk}}(c_l))\}$. It follows that $\mathcal{B}$ perfectly simulates the verification oracle and the theorem follows. $\square$

We prove the separation between CCA1.5 and IND-vCCA in Proposition 2. The proof is deferred to Appendix A.

Proposition 2. CCA1.5 $\not\Longrightarrow$ IND-vCCA

4.4 vCCA Implies FuncCPA

FuncCPA has been defined by Akavia et al. [1] to capture the security of a client-aided outsourcing protocol. These are protocols where a client outsources some computation to a server. But, since certain computations are disproportionately expensive to do homomorphically, the server may send ciphertexts to the client for re-encryption. The server may ask to perform some operation on the plaintext while it is decrypted. The FuncCPA notion is an indisitnguishability game, where the adversary gets access to a "re-encryption" algorithm, both before and after receiving the challenge ciphertext. In more details, a query consists of a ciphertext and a function (c, G_n). The oracle then decrypts the ciphertext, applies the function before returning the encryption of the result. We show that IND-vCCA security implies FuncCPA.

Definition 10 (FuncCPA [1]). *Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a* PKE *with message space $\mathcal{M}$. We say that it is funcCPA-secure with respect to a family of functions $\mathcal{G} = \{G_n : \mathcal{M} \to \mathcal{M}\}_{n \in \mathbb{N}}$ if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $\mathcal{A}$ has negligible advantage in the bit-guessing version of the indistinguishability game defined in Figure 2, with the oracle $\mathcal{O}(c, n) = \mathsf{Enc}_{\mathsf{pk}}(G_n(\mathsf{Dec}_{\mathsf{sk}}(c)))$ available in both phases.*

Theorem 4. IND-vCCA $\Longrightarrow$ FuncCPA

Proof. We define a modified FuncCPA game, where the game proceeds similarly, but in the second phase the oracle answers queries in the following way; on input (c, n) the oracle draws a random plaintext $m \leftarrow \mathcal{M}$ and sends back $c' = \mathsf{Enc}_{\mathsf{pk}}(G_n(m))$. We call this modified experiment $\mathrm{Exp}^{\mathrm{FuncCPA}^*}_{\mathcal{E},\mathcal{A}}(\lambda)$. We show that:

$$|\Pr[\mathrm{Exp}^{\mathrm{FuncCPA}}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] - \Pr[\mathrm{Exp}^{\mathrm{FuncCPA}^*}_{\mathcal{E},\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda) \tag{1}$$

Towards this we define a series of hybrid games. Let q be the number of queries made by the adversary in the second phase.
Hybrid H_q is exactly the FuncCPA experiment $\mathrm{Exp}^{\mathrm{FuncCPA}}_{\mathcal{E},\mathcal{A}}(\lambda)$.
Hybrid H_j $(j = 0, \dots q)$ proceeds similarly to H_q except that for every query in the second phase (c_i, n_i) with $i > j$, the oracle draws a random $m \leftarrow \mathcal{M}$ and sends back $\mathsf{Enc}_{\mathsf{pk}}(G_{n_i}(m))$ instead of $\mathsf{Enc}_{\mathsf{pk}}(G_{n_i}(\mathsf{Dec}_{\mathsf{sk}}(c_i)))$.

Notice that hybrid H_0 is exactly the modified experiment $\mathrm{Exp}^{\mathrm{FuncCPA}^*}_{\mathcal{E},\mathcal{A}}(\lambda)$. It follows from the standard hybrid experiment, that if equation 1 does not hold, then their exists an integer j, a polynomial $\mathsf{poly}(\lambda)$ and an adversary $\mathcal{A}$ such that for infinitely many λ:

$$|\Pr[\mathrm{Exp}^{H_{j-1}}_{\mathcal{E},\mathcal{A}}(\lambda) = 1 - \Pr[\mathrm{Exp}^{H_j}_{\mathcal{E},\mathcal{A}}(\lambda) = 1| \geq \frac{1}{\mathsf{poly}(\lambda)} \tag{2}$$

We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the IND-vCCA security of $\mathcal{E}$. Upon receiving the public-key pk, $\mathcal{B}$ draws a random j^* and runs $\mathcal{A}(\mathsf{pk})$ answering queries in the following way:

- First Phase: On input (c, n), $\mathcal{B}$ uses its decryption oracle to obtain $m = \mathsf{Dec}_{\mathsf{sk}}(c)$. $\mathcal{B}$ then computes $c' = \mathsf{Enc}_{\mathsf{pk}}(G_n(m))$ and sends c' to $\mathcal{A}$.
- Challenge: When $\mathcal{A}$ outputs (x_0, x_1), $\mathcal{B}$ draws a random $b^* \leftarrow \{0, 1\}$, computes $c^* = \mathsf{Enc}_{\mathsf{pk}}(x_{b^*})$ and send c^* to $\mathcal{A}$.
- Second Phase: On input (c_i, n_i) the ith query in the second phase:
  - If $i < j^*$, $\mathcal{B}$ answers the query as in the first phase.
  - If $i = j^*$, $\mathcal{B}$ queries c_i to its decryption oracle to obtain m_{j^*} and draws a random plaintext $m' \leftarrow \mathcal{M}$. $\mathcal{B}$ then sends $(m_0, m_1) = (m^*_j, m')$ to its challenger and sends the answer c_{chall} to $\mathcal{A}$.
  - If $i > j^*$, $\mathcal{B}$ draws a random $m \leftarrow \mathcal{M}$, computes $c' = \mathsf{Enc}_{\mathsf{pk}}(G_n(m))$ and sends c' to $\mathcal{A}$.
  - If $c_i = c_{\text{chall}}$, then $\mathcal{B}$ answers $\mathsf{Enc}_{\mathsf{pk}}(G_n(m_{j^*}))$.

When $\mathcal{A}$ finally outputs a bit b', output $b' = b^*$. It is clear that $\mathcal{B}$ simulates hybrid H_j when $c_{\text{chall}} = \mathsf{Enc}_{\mathsf{pk}}(m_0)$ and hybrid H_{j+1} when $c_{\text{chall}} = \mathsf{Enc}_{\mathsf{pk}}(m_1)$. Equation 1 follows.

Let $\mathcal{A}'$ be an adversary against the FuncCPA* security. We build an adversary $\mathcal{B}'$ against the CCA1 security of $\mathcal{E}$ that uses $\mathcal{A}'$ as a subroutine. Upon receiving the public-key pk, $\mathcal{B}'$ runs $\mathcal{A}'(\mathsf{pk})$ and answers the queries by using its decryption oracle in the first phase. In the second phase, $\mathcal{B}'$ draws a random plaintext and answers with $\mathsf{Enc}_{\mathsf{pk}}(G_n(m))$. When $\mathcal{A}'$ sends (m_0, m_1), $\mathcal{B}'$ sends it to its challenger and forwards the response to $\mathcal{A}'$. When $\mathcal{A}'$ outputs a guess b, $\mathcal{B}'$ outputs b'.

Notice that $\mathrm{Exp}^{\mathrm{FuncCPA}^*}_{\mathcal{E},\mathcal{A}'}(\lambda) = \mathrm{Exp}^{\mathrm{CCA1}}_{\mathcal{E},\mathcal{B}'}(\lambda) \leq \mathrm{Exp}^{\mathrm{IND\text{-}vCCA}}_{\mathcal{E},\mathcal{B}'}(\lambda)$. Combine with Equation 1, it follows that,

$$\mathrm{Exp}^{\mathrm{FuncCPA}}_{\mathcal{E},\mathcal{A}}(\lambda) \leq \mathrm{Exp}^{\mathrm{IND\text{-}vCCA}}_{\mathcal{E},\mathcal{B}'}(\lambda) + \mathsf{negl}(\lambda)$$

□

We now show the separation between FuncCPA and IND-vCCA. Towards this, we simply observe that bootstrapping without further precaution is incompatible with IND-vCCA, and that one of the FuncCPA schemes in [1] follows this paradigm. The proof is given in Appendix A.

Proposition 3. FuncCPA $\not\Longrightarrow$ IND-vCCA

4.5 gCCA and RCCA Imply vCCA

Finally, we show that just like HCCA, IND-vCCA is implied by both gCCA and RCCA. Both these notions are known to be unattainable for homomorphic encryption schemes [49] separating these notions from IND-vCCA. Since gCCA is known to imply RCCA [19], we only need to show that RCCA implies IND-vCCA. We use the IND-RCCA variation of the notion, as Cannetti et al. [19] have shown it to be the weakest formulation of RCCA security. We defer the definition of IND-RCCA and the proof of the theorem to Appendix A.

Theorem 5. IND-RCCA $\Longrightarrow$ IND-vCCA

5 Embedding CPA-Secure FHE into a CCA2-Secure Encryption Scheme

In this section, we define the notion of embedding FHE schemes into CCA2 secure encryption schemes following ideas from Loftus et al. [44]. We then show how to embed any CPA secure FHE scheme into a CCA2 encryption secure scheme. We give separate embedding constructions for the symmetric and asymmetric FHE, and we distinguish between schemes where the ciphertext validity check is designated verifier (requiring knowledge of a private key) or is publicly verifiable. Our constructions are *fully agnostic* to the type of FHE scheme that is to be embedded. In particular, they are compatible with bootstrapping-based FHE schemes. This is achieved by ensuring that the bootstrapping material is *not* a valid ciphertext for the CCA2 encryption scheme.

5.1 An Encryption Scheme with (Fully) Homomorphic Embedding

Informally, a (fully) homomorphic encryption scheme is embedded in a CCA2-secure encryption scheme if the ciphertexts the CCA2 scheme produces can be subdivided into two parts: a homomorphic part, that exhibits all the homomorphic properties of the underlying homomorphic encryption scheme, and a "CCA-security" part, that allows the scheme to be CCA2 secure. This means that one may be able to apply the homomorphic evaluation algorithm on the homomorphic part, but it should not be possible to create a valid ciphertext for the CCA2 secure encryption scheme from this operation.

Definition 11. *Let* $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a public-key encryption scheme, and let* $\mathsf{HE} = (\mathsf{HE.Gen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$ *be a* $\mathcal{F}$*-homomorphic encryption scheme. We say that* HE *is embedded in* $\mathcal{E}$ *if there exists an algorithm* Eval' *and* Dec' *such that, for every* $f \in \mathcal{F}$,

$$\mathsf{Dec}'_{\mathsf{sk}}(\mathsf{Eval}'_{\mathsf{pk}}(f, \{c_i\})) = f(\mathsf{Dec}'_{\mathsf{sk}}(c_1), \ldots \mathsf{Dec}'_{\mathsf{sk}}(c_l)) \tag{3}$$

with $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*, and* $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}, m_i)$ *for some plaintexts* m_i *with* $i \in \{1, \ldots, l\}$.

The definition may be relaxed so that the equality fails to hold with negligible probability. Notice that the definition directly implies that, if $\mathcal{E}$ is CCA2-secure, the output of the embedded evaluation algorithm Eval' cannot be a valid ciphertext for $\mathcal{E}$ as this would contradict its CCA2 security.

The *symmetric* variant of Definition 11 is derived in the natural way. The only subtlety is that when embedding a symmetric FHE scheme that uses bootstrapping keys, the symmetric CCA2 encryption scheme must publicly release these keys as well.

5.2 Embedding of Symmetric FHE Schemes

We start with embeddings for symmetric FHE schemes. The designated-verifier embedding uses the well-known Encrypt-then-MAC paradigm [8]. The publicly verifiable embedding follows the same structure, yet using a digital signature. The Encrypt-then-Sign construction is usually considered for public-key encryption and is not generally CCA2-secure in that setting. However, we are applying it to a *symmetric-key* encryption scheme, which allows us to build a CCA2-secure publicly verifiable scheme.

Symmetric, Designated verifier. Let $\mathcal{I} = (\mathcal{I}.\mathsf{Gen}, \mathcal{I}.\mathsf{Tag}, \mathcal{I}.\mathsf{Vfy})$ be a MAC and let $\mathsf{HE} = (\mathsf{HE.Gen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$ be a homomorphic encryption scheme. We explicitly denote the bootstrapping-key for clarity here. We define $\mathcal{E}_1 = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as follows.

- $\mathsf{Gen}(1^\lambda)$: Draw $(k_{\mathsf{HE}}, \mathsf{bk}') \leftarrow \mathsf{HE.Gen}(1^\lambda)$ and $k_{\mathcal{I}} \leftarrow \mathcal{I}.\mathsf{Gen}(1^\lambda)$ then set $k = (k_{\mathsf{HE}}, k_{\mathcal{I}})$ and $\mathsf{bk} = \mathsf{bk}'$. Output (k, bk).
- $\mathsf{Enc}(k, m)$: Output $c = (c', t)$ with $c' \leftarrow \mathsf{HE.Enc}(k_{\mathsf{HE}}, m)$ and $t \leftarrow \mathcal{I}.\mathsf{Tag}(k_{\mathcal{I}}, c')$.
- $\mathsf{Dec}(k, c)$: Parse c as (c', t). If $\mathcal{I}.\mathsf{Vfy}(k_{\mathcal{I}}, c', t) = 1$, output $\mathsf{HE.Dec}(k_{\mathsf{HE}}, c')$. Otherwise, output $\perp$.

We stress again here that the bootstrapping material from the underlying FHE scheme, if it exists, is not accompanied by a MAC tag and therefore, the bootstrapping material is *not* a valid ciphertext for the construction.

Theorem 6. *If* $\mathcal{I}$ *is a* SUF-CMA*-secure MAC, and* HE *is a* CPA*-secure FHE scheme, then the above construction is a* CCA2*-secure* PKE *with fully homomorphic embedding.*

Proof. The proof for the CCA2 security is the standard proof of security for Encrypt-then-MAC as in [8]. It is clear that HE is embedded in the construction. Indeed, for any function f, and any ciphertext $c = (c', t)$, where $(c', t) \leftarrow \mathsf{Enc}_k(m)$, we define the evaluation algorithm Eval' as, $\mathsf{Eval}'_{\mathsf{bk}}(f, c) = \mathsf{HE.Eval}_{\mathsf{bk}}(f, c')$. We then define Dec' as $\mathsf{Dec}'(k, c) = \mathsf{HE.Dec}(k_{\mathsf{HE}}, c)$. It is clear that equation 3 follows from the correctness of HE.

Symmetric, Publicly verifiable. Let $\mathcal{S} = (\mathcal{S}.\mathsf{Gen}, \mathcal{S}.\mathsf{Sign}, \mathcal{S}.\mathsf{Vfy})$ be a signature scheme and let $\mathsf{HE} = (\mathsf{HE.Gen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$ be a homomorphic encryption scheme. We define $\mathcal{E}_2 = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Vf})$ as follows.

– $\mathsf{Gen}(1^\lambda)$: Draw $(k_{\mathsf{HE}}, \mathsf{bk}') \leftarrow \mathsf{HE.Gen}(1^\lambda)$ and $(pk_{\mathcal{S}}, sk_{\mathcal{S}}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Output $(k, \mathsf{bk}) = \big((k_{\mathsf{HE}}, sk_{\mathcal{S}}, pk_{\mathcal{S}}), (\mathsf{bk}', pk_{\mathcal{S}})\big)$
– $\mathsf{Enc}(k, m)$: Output (c', t) where $c' \leftarrow \mathsf{HE.Enc}(k_{\mathsf{HE}}, m)$ and $t \leftarrow \mathcal{S}.\mathsf{Sign}(sk_{\mathcal{S}}, c')$.
– $\mathsf{Dec}(k, c)$: Parse c as (c', t). If $\mathcal{S}.\mathsf{Vfy}(pk_{\mathcal{S}}, c', t) = 1$, output $\mathsf{Dec}(k_{\mathsf{HE}}, c')$. Otherwise, output $\perp$.
– $\mathsf{Vf}(\mathsf{bk}, c)$: Parse c as (c', t). Output $\mathcal{S}.\mathsf{Vfy}(pk_{\mathcal{S}}, c', t)$.

Corollary 1. *If* Sig *is a* SUF-CMA-*secure signature scheme, and* HE *is a* CPA-*secure FHE scheme, then the above construction is a publicly verifiable* CCA2-*secure symmetric-key encryption with fully homomorphic embedding.*

Proof. The proof is essentially the same as for the designated-verifier construction. Using a signature scheme instead of a MAC can be seen as giving a verification oracle to the adversary. It is a well known result (see e.g. [16]) that SUF-CMA security is equivalent to SUF-CMA security with verfication oracle. It is clear that the Vf algorithm performs the validity check correctly without the need for the secret key.

5.3 Embedding of Asymmetric FHE Schemes

We now give embeddings for asymmetric FHE schemes. Both embeddings are based on the principle of double encryption. In the publicly verifiable case, we use the Naor-Yung approach [47]. However, in the designated-verifier case, we are able to forgo the use of a NIZK and obtain a more efficient encryption scheme.

Asymmetric, Designated verifier. Let $\mathsf{HE} = (\mathsf{HE.Gen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$ be a homomorphic encryption scheme with message space $\mathcal{M}$. We write COIN to denote the randomness space of HE. Let $\mathcal{E} = (\mathcal{E}.\mathsf{Gen}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec})$ be a public-key encryption scheme with message space $\mathcal{M} \times \mathsf{COIN}$. We define $\mathcal{E}_3 = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as follows.

– $\mathsf{Gen}(1^\lambda)$: Draw $(\mathsf{pk}_0, \mathsf{sk}_0, \mathsf{bk}) \leftarrow \mathsf{HE.Gen}(1^\lambda)$ and $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathcal{E}.\mathsf{Gen}(1^\lambda)$. Output $(\mathsf{pk}, \mathsf{sk}) = \big((\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{bk}), \mathsf{sk}_1\big)$
– $\mathsf{Enc}(\mathsf{pk}, m)$: Sample a random value $r \xleftarrow{\$} \mathsf{COIN}$. Output $c = (c_0, c_1)$ where $c_0 \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_0, m; r)$ and $c_1 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_1, m \| r)$.

– $\mathsf{Dec}(\mathsf{sk}, c, t)$: Parse c as (c_0, c_1). Compute and parse $m \| r \leftarrow \mathcal{E}.\mathsf{Dec}(\mathsf{sk}_1, c_1)$. If $\mathsf{HE}.\mathsf{Enc}(\mathsf{pk}_0, m; r) = c_0$, output m. Otherwise, output $\perp$.

Theorem 7. *If $\mathcal{E}$ is a* CCA2*-secure* PKE*, and* HE *is a* CPA*-secure FHE scheme, then the above construction is a* CCA2*-secure* PKE *with fully homomorphic embedding.*

Proof. HE is embedded in the construction in a similar manner as with the previous constructions. To prove the CCA2-security, we define a series of games G_0, G_1, G_2, G_3. We denote with W_i the event that the adversary $\mathcal{A}$ outputs 1 at the end of the game G_i.

Let CCA2-b be the CCA2 game where the challenger picks the bit b. Let game G_0 be the CCA2-0 game. Let game G_1 be the same game as G_0, but the challenge ciphertext $c^* = (c_0^*, c_1^*)$ is now computed differently. The ciphertext c_1^* is computed by encrypting $m_1 \| r'$ instead of $m_0 \| r$, where r' is a new random coin $r' \leftarrow \mathsf{COIN}$ that may be different from the coin used to produce c_0^*. Additionally, the challenger rejects decryption queries of the form (c_0', c_1^*).

Notice that in game G_0, there exists no valid ciphertext (c_0', c_1^*) with $c_0' \neq c_0^*$. Indeed, m_b, r, pk_{HE} fully determine $c_0 = \mathsf{HE}.\mathsf{Enc}(pk_{\mathsf{HE}}, m_0; r)$. This is why in game G_1, we require the challenger to reject any query of that form. This can be thought of as generating simulated proofs when using a NIZK.

Let $\mathcal{A}_0$ be an adversary that distinguishes between G_0 and G_1. Then we can build an adversary $\mathcal{B}_0$ that breaks the CCA2 security of $\mathcal{E}$. Upon receiving the public-key $pk_{\mathcal{E}}$, $\mathcal{B}_0$ generates the keys $(pk_{\mathsf{HE}}, sk_{\mathsf{HE}}, \mathsf{bk}) \leftarrow \mathsf{HE}.\mathsf{Gen}(1^\lambda)$ for the HE scheme and sets $\mathsf{pk} = (pk_{\mathsf{HE}}, pk_{\mathcal{E}}, \mathsf{bk})$. It then runs $\mathcal{A}_0(\mathsf{pk})$ and answers queries as follows.

– **Decryption queries:** In the first phase, $\mathcal{B}_0$ answers decryption queries of the form $c = (c_0, c_1)$ by forwarding c_1 to its decryption oracle. $\mathcal{B}$ parses the answer as $m \| r$ and computes $c' \leftarrow \mathsf{Enc}(pk_{\mathsf{HE}}, m; r)$. If $c' = c_0$, it returns m and $\perp$ otherwise. In the second phase, $\mathcal{B}_0$ does the same, but answers $\perp$ to any query of the form (c_0', c_1^*) for any c_0'.
– **Challenge phase:** When $\mathcal{B}_0$ receives the challenge plaintexts (m_0, m_1), it draws two random coins $r, r' \leftarrow \mathsf{COIN}$ and sends $(m_0 \| r', m_1 \| r')$ to its challenger. It receives the challenge ciphertext $c_1^* = \mathcal{E}.\mathsf{Enc}(pk_{\mathcal{E}}, m_b \| r')$ for some unknown b. $\mathcal{B}_0$ then computes $c_0^* = \mathsf{HE}.\mathsf{Enc}(pk_{\mathsf{HE}}, m_0; r)$, sets $c^* = (c_0^*, c_1^*)$ and returns (c_0^*, c_1^*).

Finally, when $\mathcal{A}_0$ outputs a guess $\hat{b}$, $\mathcal{B}_0$ does the same. $\mathcal{B}_0$ perfectly simulates game G_0 when $b = 0$, and game G_1 when $b = 1$. It follows that,

$$|\Pr[W_0] - \Pr[W_1]| \leq \mathsf{Adv}^{\mathrm{CCA2}}_{\mathcal{B}_0, \mathcal{E}}(\lambda) \tag{4}$$

Let game G_2 be the same game as G_1, but c_0^* is computed by encrypting m_1 instead of m_0. Let $\mathcal{A}_1$ be an adversary that distinguishes between G_1 and G_2. Then we can build an adversary $\mathcal{B}_1$ that breaks the CPA security of HE. Upon receiving the public-key pk_{HE}, $\mathcal{B}_1$ generates the keys $(pk_{\mathcal{E}}, sk_{\mathcal{E}}) \leftarrow \mathcal{E}.\mathsf{Gen}(1^\lambda)$ for the $\mathcal{E}$ scheme and sets $\mathsf{pk} = (pk_{\mathsf{HE}}, pk_{\mathcal{E}})$. It then runs $\mathcal{A}_1(\mathsf{pk})$ and answers queries as follows.

– **Decryption queries:** In the first phase, it answers decryption queries by using $sk_{\mathcal{E}}$ to run the Dec algorithm. In the second phase, it does the same but rejects any query of the form (c_0', c_1^*).
– **Challenge phase:** When $\mathcal{B}_1$ receives the challenge plaintexts (m_0, m_1), it sends (m_0, m_1) to its challenger. It then receives the ciphertext $c_0^* = \mathsf{HE.Enc}(pk_{\mathsf{HE}}, m_b; r)$ for some unknown b and r. $\mathcal{B}_1$ then draws a random coin $r' \leftarrow \mathsf{COIN}$, computes $c_1^* = \mathcal{E}.\mathsf{Enc}(pk_{\mathcal{E}}, m_1 \| r')$ and returns $c^* = (c_0^*, c_1^*)$.

Finally, when $\mathcal{A}_1$ outputs a guess $\hat{b}$, $\mathcal{B}_1$ does the same. $\mathcal{B}_1$ simulates game G_1 when $b = 0$, and simulates game G_2 when $b = 1$. It follows that,

$$|\Pr[W_1] - \Pr[W_2]| \leq \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{B}_1, \mathsf{HE}}(\lambda) \tag{5}$$

Game G_3 is the same as game G_2 except we return to computing c_1^* by using the same coin r as the one used in the encryption of c_0^*, i.e. we have $c_0^* = \mathsf{HE.Enc}(pk_{\mathsf{HE}}, m_1; r)$ and $c_1^* = \mathcal{E}.\mathsf{Enc}(pk_{\mathcal{E}}, m_1 \| r)$. We can build an adversary $\mathcal{B}_2$ against the CCA2 security of $\mathcal{E}$ from a distinguisher between G_2 and G_3 in the same manner as we built $\mathcal{B}_0$. It follows that,

$$|\Pr[W_2] - \Pr[W_3]| \leq \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{B}_2, \mathsf{HE}}(\lambda) \tag{6}$$

Remember that G_0 is the CCA2-0 game, and notice that G_3 is exactly the CCA2-1 game. The theorem follows from equations 4, 5 and 6.

□

For completeness, we give the construction for the publicly verifiable, asymmetric case which simply uses the Naor-Yung paradigm [47]. The proof of security of the CCA2 variant can be found in [52]. This type of construction has already been used for FHE in [14, 20].

Asymmetric, Publicly verifiable. Let $\mathsf{HE} = (\mathsf{HE.Gen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$ be a homomorphic encryption scheme and $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{P}, \mathsf{Vf}', \mathsf{Sim})$ be a simulation-sound NIZK for the language of double encryptions We define $\mathcal{E}_3 = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Vf})$ as follows.

– $\mathsf{Gen}(1^\lambda)$: Draw $(\mathsf{pk}_0, \mathsf{sk}_0, \mathsf{bk}_0) \leftarrow \mathsf{HE.Gen}(1^\lambda)$, $(\mathsf{pk}_1, \mathsf{sk}_1, \mathsf{bk}_1) \leftarrow \mathsf{HE.Gen}(1^\lambda)$ and $\sigma \leftarrow \mathsf{Setup}$. Output $(\mathsf{pk}, \mathsf{sk}) = \big((\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{bk}, \sigma), (\mathsf{sk}_0)\big)$
– $\mathsf{Enc}(\mathsf{pk}, m)$: Compute $c_0 \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_0, m; r_0)$ and $c_1 \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_1, m; r_1)$. Output (c_0, c_1, π) where $\pi \leftarrow \mathsf{P}(c_0, c_1, m, r_0, r_1)$.
– $\mathsf{Dec}(\mathsf{sk}, c)$: Parse c as (c_0, c_1, π). If $\mathsf{Vf}(\pi, c_0, c_1)$, output $m \leftarrow \mathsf{HE.Dec}(\mathsf{sk}_0, c_0)$. Otherwise, output $\perp$.
– $\mathsf{Vf}(\mathsf{pk}, c)$: Parse c as (c_0, c_1, π). Output $\mathsf{Vf}'(\pi, c_0, c_1)$.

5.4 On Approximate FHE

We highlight again here that special care needs to be taken when embedding approximate FHE schemes such as the CKKS scheme [24]. Indeed, the CCA2 transforms we use, i.e., Encrypt-then-MAC and the Naor-Yung double encryption paradigm assume the encryption scheme to be passively secure. CPA security does not adequately capture that for approximate FHE schemes as shown by Li and Micciancio in [43]. It is easy to see that the CPA^D attack that they introduce is still effective even after the CCA2 transformation. This is solved by using a CPA^D secure encryption scheme as a starting point for the transformation when considering approximate encryption schemes.

6 Building vCCA-Secure FHE

We now show how to transform a CCA2 secure encryption scheme with homomorphic embedding into a IND-vCCA secure FHE scheme by using a SNARK. We give two variations, depending on if the encryption scheme is publicly verifiable or designated verifier. Both can accommodate the use of a symmetric embedded encryption scheme or an asymmetric one. We will use the public-key notation but the symmetric case is built in a similar manner. The construction uses the CCA2-secure encryption scheme to ensure the security of the output of the encryption algorithm. We then use the SNARK to prove the correct execution of the evaluation algorithm. This links the evaluated ciphertext to the input ciphertext that have been used to generate it. This allows to reduce any attack on the FHE scheme to the CCA2 scheme. The extraction algorithm mandated by the IND-vCCA security is then simply the witness extractor of the SNARK.

More precisely, given a CCA2 encryption scheme with (fully) homomorphic embedding $\mathcal{E}$, we build an IND-vCCA secure FHE scheme as follows. The encryption algorithm is just the encryption algorithm of $\mathcal{E}$. The evaluation algorithm uses the evaluation algorithm embedded in $\mathcal{E}$, and uses the SNARK to prove that all the inputs where valid i.e. inputs that pass the validity check of $\mathcal{E}$, or in the case of repeated evaluation, ciphertexts that have a valid proof attached to them. The decryption algorithm then checks the proof before using the decryption algorithm of $\mathcal{E}$, if the proof is valid.

6.1 Constructions

Designated verifier case. Let $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Vfy}, \Pi.\mathsf{Sim})$ be a SNARK and let $\mathcal{E} = (\mathcal{E}.\mathsf{Gen}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec})$ be a designated verifier encryption scheme with homomorphic embedding. Let Eval', Dec' and Vf be the embedded evaluation algorithm, decryption algorithm and the canonical verification algorithm for $\mathcal{E}$ respectively. Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathcal{E}.\mathsf{Gen}(1^\lambda)$ be a key pair for the $\mathcal{E}$. We define the language L_1 as:

$$L_1 = \left\{(\{c_i\}, \hat{c}) \mid \exists f \in \mathcal{F}',\ \hat{c} = \mathsf{Eval}'(\mathsf{pk}', f, \{c_i\})\right\}$$

We then define $\mathsf{HE}_{\text{desig}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ as follows.

- $\mathsf{Gen}(1^\lambda)$: Draw $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathcal{E}.\mathsf{Gen}(1^\lambda)$, and $\sigma \leftarrow \mathit{\Pi}.\mathsf{Setup}(1^\lambda, R_1)$, where R_1 is the relation corresponding to L_1. Output $(\mathsf{pk}, \mathsf{sk}) = \big((\mathsf{pk}', \sigma), (\mathsf{sk}', \sigma)\big)$.
- $\mathsf{Enc}(\mathsf{pk}, m)$: Output $c \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}', m)$.
- $\mathsf{Eval}(\mathsf{pk}, f, \{c_i\})$: Output $(\hat{c}, \pi, \{c_i\})$ where $\hat{c} \leftarrow \mathsf{Eval}'(\mathsf{pk}', f, \{c_i\})$ and $\pi \leftarrow \mathit{\Pi}.\mathsf{Prove}(\sigma, f, \{c_i\}, \hat{c})$.
- $\mathsf{Dec}(\mathsf{sk}, \hat{c}, \pi, \{c_i\})$: If $\mathit{\Pi}.\mathsf{Vfy}(\sigma, f, \{c_i\}, \hat{c}, \pi) = 1$ and if for all i, $\mathsf{Vf}(\mathsf{sk}', c_i) = 1$, then output $\mathsf{Dec}'(\mathsf{sk}', \hat{c})$. Otherwise output $\perp$.

The publicly verifiable variation of the construction is mostly the same as in the designated verifier case. The only difference is that we are able to make the evaluator do the validity check of the input ciphertexts. This allows the scheme to achieve better compactness as we do not need to send the fresh ciphertexts as part of the statement for the SNARK proof.

Publicly verifiable case. Let $\mathit{\Pi} = (\mathit{\Pi}.\mathsf{Setup}, \mathit{\Pi}.\mathsf{Prove}, \mathit{\Pi}.\mathsf{Vfy}, \mathit{\Pi}.\mathsf{Sim})$ be a SNARK for the language L_2 defined bellow. Let $\mathcal{E} = (\mathcal{E}.\mathsf{Gen}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec})$ be a publicly verifiable encryption scheme with homomorphic embedding. Let Eval', Dec' and Vf be the embedded evaluation algorithm, decryption algorithm and the verification algorithm for $\mathcal{E}$ respectively.

$$L_2 = \left\{ \hat{c} \;\middle|\; \exists (f, \{c_i\}) \begin{array}{l} \hat{c} = \mathsf{HE.Eval}(\mathsf{pk}', f, \{c_i\}) \\ \forall i \ \mathsf{Vf}(\mathsf{pk}', c_i) = 1 \end{array} \right\}$$

We then define $\mathsf{HE}_{\mathrm{pub}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ as follows.

- $\mathsf{Gen}(1^\lambda)$: Draw $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathcal{E}.\mathsf{Gen}(1^\lambda)$, and $\sigma \leftarrow \mathit{\Pi}.\mathsf{Setup}(1^\lambda, R_2)$, where R_2 is the relation corresponding to L_2. Output $(\mathsf{pk}, \mathsf{sk}) = \big((\mathsf{pk}', \sigma), (\mathsf{sk}', \sigma)\big)$
- $\mathsf{Enc}(\mathsf{pk}, m)$: Output $c \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}', m)$
- $\mathsf{Eval}(\mathsf{pk}, f, \{c_i\})$: Output $(\hat{c}, \pi)$, where $\hat{c} \leftarrow \mathsf{Eval}'(\mathsf{pk}', f, \{c_i\})$ and $\pi \leftarrow$ $\mathit{\Pi}.\mathsf{Prove}(\sigma, f, \{c_i\}, \hat{c})$.
- $\mathsf{Dec}(\mathsf{sk}, \hat{c}, \mathit{\Pi})$: If $\mathit{\Pi}.\mathsf{Vfy}(\sigma, \hat{c}, \mathit{\Pi})$, then output $\mathsf{Dec}'(\mathsf{sk}, \hat{c})$.
  Otherwise output $\perp$.

Remark 3. We write both constructions here for the case of a single homomorphic evaluation. In the case of repeated homomorphic evaluations, the evaluator has to prove that it is using either valid fresh ciphertexts, or valid evaluated ciphertexts, i.e. evaluated ciphertexts that have a valid proof attached to them. More precisely, consider a tree, where the evaluated ciphertext is the root, the edges are the consecutive evaluations, the nodes are the intermediate evaluated ciphertexts, and the leaves are the freshly encrypted ciphertexts. Then, in the designated-verifier case, the evaluator must prove that every node has a valid proof attached, and output every leaf for the decrypting party to verify the validity. The publicly verifiable case is the same, but the evaluator does not need to output the leaves. Instead it proves that they are valid ciphertexts with respect to $\mathcal{E}$. This is similar to what is discussed in [14] for targeted malleability.

6.2 Security Proof

We present the security of both constructions in Theorem 8. The security proofs for both are very similar. We give the security proof for $\mathsf{HE}_{\text{desig}}$ and explain afterwards the few differences regarding HE_{pub}.

Theorem 8. *Let* HE *be a* CPA *secure homomorphic encryption scheme embedded in* $\mathcal{E} = (\mathcal{E}.\mathsf{Gen}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec})$ *with embedded evaluation and decryption algorithm* Eval' *and* Dec'. *Then, if* $\mathcal{E}$ *is* CCA2*-secure, and* Π *is a black-box weak SE SNARK for* L_1 *(respectively for* L_2*), then* HE_{desig} *(respectively* HE_{pub}*) is* IND-vCCA*-secure. More precisely, for every* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *adversary* $\mathcal{B}$ *such that,*

$$\mathsf{Adv}^{\text{IND-vCCA}}_{\mathcal{A},\mathsf{HE}_{desig}}(\lambda) \leq \mathsf{Adv}^{\text{CCA2}}_{\mathcal{B},\mathcal{E}}(\lambda) + \mathsf{negl}(\lambda)$$
$$\mathsf{Adv}^{\text{IND-vCCA}}_{\mathcal{A},\mathsf{HE}_{pub}}(\lambda) \leq \mathsf{Adv}^{\text{CCA2}}_{\mathcal{B},\mathcal{E}}(\lambda) + \mathsf{negl}(\lambda)$$

Proof. We first construct the extraction algorithm for $\mathsf{HE}_{\text{desig}}$. Let χ_1 be the witness extraction algorithm of the underlying SNARK. Recall that the evaluation algorithm outputs messages of the form $(\hat{c}, \pi, \{c_i\})$. Let aux be the extraction trapdoor τ_{ext}. The extraction algorithm $\mathsf{Extract}_1(\text{aux}, \hat{c}, \pi, \{c_i\})$ uses the witness extraction χ_1 to recover f and then outputs $(f, \{c_i\})$.

We now show that this is a sound extraction algorithm. The decryption algorithm Dec first checks the validity of the proof π before decrypting the ciphertext. It follows from the black-box weak SE of the SNARK that if the proof verifies:

$$\Pr[\hat{c} = \mathsf{Eval}(f, \{c_i\})] \geq 1 - \mathsf{negl}(\lambda)$$

The correctness of $\mathcal{E}$ then implies that $\mathsf{Dec}'_{sk}(\hat{c}) = f(\mathsf{Dec}'_{\mathsf{sk}}(c_1), \ldots, \mathsf{Dec}'_{\mathsf{sk}}(c_l))$. Furthermore, the decryption algorithm Dec returns $\perp$ if one of the ciphertext c_i is invalid; recall that in that case, $f(\mathsf{Dec}'_{\mathsf{sk}}(c_1), \ldots, \mathsf{Dec}'_{\mathsf{sk}}(c_l)) = \perp$. It follows that $\mathsf{Dec}_{sk}(\hat{c}) = f(\mathsf{Dec}_{\mathsf{sk}}(c_1), \ldots, \mathsf{Dec}_{\mathsf{sk}}(c_l))$ with overwhelming probability so $\mathsf{Extract}_1$ is sound.

Let $\mathcal{A}$ be a PPT adversary against the IND-vCCA security of $\mathsf{HE}_{\text{desig}}$. We build an adversary $\mathcal{B}$ against the CCA2 security of $\mathcal{E}$ that uses $\mathcal{A}$ as a subroutine. Upon receiving the public-key pk' for $\mathcal{E}$, $\mathcal{B}$ runs $\sigma \leftarrow \Pi.\mathsf{Setup}(1^\lambda, R_1)$ and sets $\mathsf{pk} = (\mathsf{pk}', \sigma)$. It then runs $\mathcal{A}(\mathsf{pk})$ and answers queries as follows.

- In the first phase, when a decryption query c comes in, $\mathcal{B}$ computes $(f, c_1, \ldots, c_l) \leftarrow \mathsf{Extract}_1(c)$. It queries every $c_i \in (c_1, \ldots, c_l)$ to its decryption oracle and receives answers m_i. It then returns $f(m_1, \ldots, m_l)$, or $\perp$ if there is a j such that $m_j = \perp$.
- The challenge pair (m_0, m_1) are forwarded to the challenger and $\mathcal{B}$ gives the response c^* to $\mathcal{A}$.
- The second phase is handled as the first phase, but $\mathcal{B}$ returns $\perp$ if $c^* \in (c_1, \ldots, c_l)$. Otherwise, it uses its decryption oracle as in the first phase to answer the queries.

Recall that Enc is the same algorithm as $\mathcal{E}.\mathsf{Enc}$, so no further work is required when transferring c^* to $\mathcal{A}$ from the challenger. Additionally, since $\mathsf{Extract}_1$ is sound, $\mathcal{B}$ perfectly answers queries in the second phase with overwhelming probability. Therefore, when $\mathcal{A}$ outputs a guess b, $\mathcal{B}$ does the same. It follows that,

$$\mathsf{Adv}^{\text{IND-vCCA}}_{\mathcal{A},\mathsf{HE}_{\text{desig}}}(\lambda) \leq \mathsf{Adv}^{\text{CCA2}}_{\mathcal{B},\mathcal{E}}(\lambda) + \mathsf{negl}(\lambda),$$

where the second term comes from the black-box weak SE property of the SNARK.

The security for HE_{pub} is proven similarly. The extraction algorithm $\mathsf{Extract}_2$ uses the witness extractor of the underlying SNARK to obtain $(f, c_1, \ldots, c_l)$ directly. The soundness is then proven in the same way, but the argument is simpler as the validity check of fresh ciphertexts is already included in the proof. □

6.3 CCA1 Security

We also highlight that if CCA1 security is sufficient, our transformation would work from weaker assumptions. The encryption scheme with homomorphic embedding only needs to be CCA1 secure instead of CCA2, and, more importantly, it suffices that the SNARK is white-box extractable. This may be important in practice as such SNARKs are more efficient and significantly easier to achieve.

Theorem 9. *Let $\mathcal{E} = (\mathcal{E}.\mathsf{Gen}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec})$ be as defined in Theorem 8. Then, if $\mathcal{E}$ is* CCA1*-secure, and Π is a white-box weak SE SNARK for L_1 (respectively for L_2), then HE_{desig} (respectively HE_{pub}) is* CCA1*-secure. More precisely, for every* PPT *adversary $\mathcal{A}$, there exists a* PPT *adversary $\mathcal{B}$ such that,*

$$\mathsf{Adv}^{\text{CCA1}}_{\mathcal{A},\mathsf{HE}_{desig}}(\lambda) \leq \mathsf{Adv}^{\text{CCA1}}_{\mathcal{B},\mathcal{E}}(\lambda) + \mathsf{negl}(\lambda)$$
$$\mathsf{Adv}^{\text{CCA1}}_{\mathcal{A},\mathsf{HE}_{pub}}(\lambda) \leq \mathsf{Adv}^{\text{CCA1}}_{\mathcal{B},\mathcal{E}}(\lambda) + \mathsf{negl}(\lambda)$$

Proof. The proof is very similar to the proof of Theorem 8. The main difference is that we do not need to build the extraction algorithm for the IND-vCCA notion. Decryption queries are answered as in Theorem 8, but we use the witness extraction algorithm directly to obtain $(f, c_1, \ldots, c_l)$.

7 Conclusion and Future Work

In this work, we introduce the new IND-vCCA security notion, which is strictly stronger than CCA1 security and show that it is achievable for FHE schemes. We give a general transformation that allows to build IND-vCCA secure schemes from any CPA secure FHE scheme. This also gives the *first* CCA1 secure FHE scheme based on bootstrapping techniques.

We have also shown that to obtain CCA1 security, it is possible to relax the requirements of our transformation by embedding the FHE scheme into a CCA1 secure encryption scheme and using only a white-box SE SNARK. Furthermore, consider *circuit privacy* [37], which asks that the homomorphic evaluation algorithm does not reveal information on the evaluated function. IND-vCCA as defined here seems to be a contradiction to circuit privacy. However, we expect that a CCA1 version of our transformation should be able to achieve some form of circuit privacy. It may also be interesting to explore definitions of circuit privacy for schemes that achieve security beyond CCA1.

We have left our transformation fully generic. Another avenue would be to look at specific FHE schemes, embedding methods and SNARKs to select those that exhibit the best compatibility. Even further, Ganesh et al. [36] have designed a SNARK with the specific intention of allowing for better compatibility with FHE schemes. It would be interesting to do this for all three elements of our transformation together.

Finally, while we have shown that vCCA is the strongest notion for FHE schemes among those that we have considered, we have not shown that it is the strongest possible. Is there a stronger and meaningful notion than vCCA that is achievable by FHE schemes?

References

1. Akavia, A., Gentry, C., Halevi, S., Vald, M.: Achievable CCA2 relaxation for homomorphic encryption. In: TCC. pp. 70–99. Springer (2022)
2. Akavia, A., Vald, M.: On the privacy of protocols based on CPA-secure homomorphic encryption. Cryptology ePrint Archive (2021)
3. An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: EUROCRYPT 2002. pp. 83–107. Springer (2002)
4. Armknecht, F., Katzenbeisser, S., Peter, A.: Group homomorphic encryption: characterizations, impossibility results, and applications. Designs, codes and cryptography **67**, 209–232 (2013)
5. Atapoor, S., Baghery, K., Pereira, H.V., Spiessens, J.: Verifiable FHE via Lattice-based SNARKs. Cryptology ePrint Archive (2024)
6. Baghery, K., Kohlweiss, M., Siim, J., Volkhov, M.: Another look at extraction and randomization of Groth's zk-SNARK. In: FC 2021. pp. 457–475. Springer (2021)
7. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: CRYPTO'98. pp. 26–45. Springer (1998)
8. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: ASIACRYPT 2000. pp. 531–545. Springer (2000)
9. Bellare, M., Palacio, A.: Towards plaintext-aware public-key encryption without random oracles. In: ASIACRYPT 2004. pp. 48–62. Springer (2004)
10. Biasse, J.F., Fieker, C.: Subexponential class group and unit group computation in large degree number fields. LMS Journal of Computation and Mathematics **17**(A), 385–403 (2014)
11. Biasse, J.F., Song, F.: Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In: SODA16. pp. 893–902. SIAM (2016)

12. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: ITCS 2012. pp. 326–349 (2012)
13. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS# 1. In: CRYPTO'98. pp. 1–12. Springer (1998)
14. Boneh, D., Segev, G., Waters, B.: Targeted malleability: homomorphic encryption for restricted computations. In: ITCS 2012. pp. 350–366 (2012)
15. Boneh, D., Shen, E., Waters, B.: Strongly unforgeable signatures based on computational Diffie-Hellman. In: PKC 2006. pp. 229–240. Springer (2006)
16. Boneh, D., Shoup, V.: A graduate course in applied cryptography. Draft 0.6 (2023)
17. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: ITCS '12. p. 309–325 (2012)
18. Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G.N., Rothblum, R.D.: Fiat-Shamir from simpler assumptions. Cryptology ePrint Archive (2018)
19. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: CRYPTO 2003. pp. 565–582. Springer (2003)
20. Canetti, R., Raghuraman, S., Richelson, S., Vaikuntanathan, V.: Chosen-ciphertext secure fully homomorphic encryption. In: PKC 2017. pp. 213–240. Springer (2017)
21. de Castro, L., Peikert, C.: Functional Commitments for All Functions, with Transparent Setup and from SIS. In: EUROCRYPT 2023. pp. 287–320. Springer (2023)
22. Checri, M., Sirdey, R., Boudguiga, A., Bultel, J.P.: On the practical cpad security of "exact" and threshold fhe schemes and libraries. Cryptology ePrint Archive, Paper 2024/116
23. Chen, H., Laine, K., Rindal, P.: Fast private set intersection from homomorphic encryption. In: ACM SIGSAC 2017. pp. 1243–1255 (2017)
24. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: ASIACRYPT 2017. pp. 409–437. Springer (2017)
25. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (2020)
26. Chillotti, I., Gama, N., Goubin, L.: Attacking FHE-based applications by software fault injections. Cryptology ePrint Archive (2016)
27. Cramer, R., Ducas, L., Peikert, C., Regev, O.: Recovering short generators of principal ideals in cyclotomic rings. In: EUROCRYPT 2016. pp. 559–585. Springer (2016)
28. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: CRYPTO'98. pp. 13–25. Springer (1998)
29. Das, A., Dutta, S., Adhikari, A.: Indistinguishability against chosen ciphertext verification attack revisited: The complete picture. In: ProvSec 2013. pp. 104–120. Springer (2013)
30. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. In: ACM STOC 1991. pp. 542–552 (1991)
31. Ducas, L., Micciancio, D.: Fhew: bootstrapping homomorphic encryption in less than a second. In: EUROCRYPT 2015. pp. 617–640. Springer (2015)
32. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory **31**(4), 469–472 (1985)
33. Fauzi, P., Hovd, M.N., Raddum, H.: On the IND-CCA1 security of FHE schemes. Cryptography **6**(1), 13 (2022)
34. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of cryptology **26**, 80–101 (2013)

35. Ganesh, C., Kondi, Y., Orlandi, C., Pancholi, M., Takahashi, A., Tschudi, D.: Witness-succinct universally-composable SNARKs. In: EUROCRYPT 2023. pp. 315–346. Springer (2023)
36. Ganesh, C., Nitulescu, A., Soria-Vazquez, E.: Rinocchio: Snarks for ring arithmetic. Journal of Cryptology **36**(4), 41 (2023)
37. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: ACM STOC 2009. p. 169–178. STOC '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1536414.1536440
38. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: CRYPTO 2013. pp. 75–92. Springer (2013)
39. Geva, R., Gusev, A., Polyakov, Y., Liram, L., Rosolio, O., Alexandru, A., Genise, N., Blatt, M., Duchin, Z., Waissengrin, B., et al.: Collaborative privacy-preserving analysis of oncological data using multiparty homomorphic encryption. Proceedings of the National Academy of Sciences **120**(33), e2304415120 (2023)
40. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: ASIACRYPT 2006. pp. 444–459. Springer (2006)
41. Groth, J.: On the size of pairing-based non-interactive arguments. In: EUROCRYPT 2016. pp. 305–326. Springer (2016)
42. Krohn, M.N.: On the Definitions of Cryptographic Security: Chosen Ciphertext Attack Revisited. Ph.D. thesis, Citeseer (1999)
43. Li, B., Micciancio, D.: On the security of homomorphic encryption on approximate numbers. In: EUROCRYPT 2021. pp. 648–677. Springer (2021)
44. Loftus, J., May, A., Smart, N.P., Vercauteren, F.: On CCA-secure somewhat homomorphic encryption. In: SAC 2011. pp. 55–72. Springer (2012)
45. Lu, W.j., Huang, Z., Hong, C., Ma, Y., Qu, H.: Pegasus: bridging polynomial and non-polynomial evaluations in homomorphic encryption. In: S&P 2021. pp. 1057–1073. IEEE (2021)
46. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key fhe. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016. pp. 735–763. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
47. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: ACM STOC 1990. pp. 427–437 (1990)
48. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: EUROCRYPT '99. pp. 223–238. Springer (1999)
49. Prabhakaran, M., Rosulek, M.: Homomorphic encryption with cca security. In: ICALP. pp. 667–678. Springer (2008)
50. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: CRYPTO '91. pp. 433–444. Springer (1991)
51. Rivest, R.L., Adleman, L., Dertouzos, M.L., et al.: On data banks and privacy homomorphisms. Foundations of secure computation **4**(11), 169–180 (1978)
52. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS 1999. pp. 543–553. IEEE (1999)
53. Viand, A.: Useable Fully Homomorphic Encryption. Ph.D. thesis, ETH Zurich (2023)
54. Viand, A., Knabenhans, C., Hithnawi, A.: Verifiable fully homomorphic encryption. arXiv preprint arXiv:2301.07041 (2023)
55. Wang, B., Wang, X., Xue, R.: CCA1 secure FHE from PIO, revisited. Cybersecurity **1**(1), 1–8 (2018)

56. Yasuda, S., Kitagawa, F., Tanaka, K.: Constructions for the IND-CCA1 secure fully homomorphic encryption. Mathematical Modelling for Next-Generation Cryptography: CREST Crypto-Math Project pp. 331–347 (2018)
57. Zhang, Z., Plantard, T., Susilo, W.: Reaction attack on outsourced computing with fully homomorphic encryption schemes. In: ICISC 2011. pp. 419–436. Springer (2012)

A Further Proofs

A.1 Proof of Proposition 1

Proof. Recall that unlinkability [49, Definition 3] is defined as a distinguishing game. An adversary $\mathcal{A}$ that has access to the public-key and a decryption oracle, chooses a ciphertext c and a function f, and must distinguish between an encryption of $c_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(f(\mathsf{Dec}_{\mathsf{sk}}(c)))$ and the evaluation of the ciphertext $c_2 \leftarrow \mathsf{Eval}_{\mathsf{pk}}(f, c)$.

If c_1 is indistinguishable from c_2, then $\mathsf{Extract}(\mathrm{aux}, c_1)$ and $\mathsf{Extract}(\mathrm{aux}, c_2)$ must be indistinguishable. It follows from the soundness of the extraction algorithm that the statement $\Phi_1 \leftarrow \mathsf{Extract}(\mathrm{aux}, c_1)$ is in $\Phi_1 \in \{\perp, (\mathrm{Id}, c_1)\}$ depending on the extraction algorithm. However, for $\Phi_2 \leftarrow \mathsf{Extract}(\mathrm{aux}, c_1)$, we have $\Phi_2 = (f, c_2)$ so they are trivially distinguishable. It follows that IND-vCCA security is a contradiction to the unlinkability.

In [49, Section 5], Prabhakaran et al. give a construction and show that it is both HCCA and unlinkable. The above contradiction implies that the construction cannot be IND-vCCA and the proposition follows. □

A.2 Proof of Proposition 2

We revisit a construction given in [29] that is proven to be CCA1.5 and prove that it is not IND-vCCA secure.

Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be the Paillier encryption scheme [48]. Consider $\mathcal{E}' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ as defined in Figure 5. Das et al. have shown $\mathcal{E}'$ to be CCA1.5 secure [29, Lemma 6]. We now show that it is not IND-vCCA secure.

$\mathsf{Gen}'(1^\lambda)$	$\mathsf{Enc}'(m)$	$\mathsf{Dec}'(c')$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$	$c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$	Parse $c' = (b \| c), b \in \{0, 1\}$
return $(\mathsf{pk}, \mathsf{sk})$	**return** $c' \leftarrow (0 \| c)$	**if** $b = 0,$ **return** $\mathsf{Dec}_{\mathsf{sk}}(c)$
		if $b = 1,$ **return** $\perp$

Fig. 5. CCA1.5 construction from the Paillier encryption scheme.

Proof. Let $\mathcal{E}'$ be the encryption scheme as defined in Figure 5. We show that $\mathcal{E}'$ is not IND-vCCA secure. To break the IND-vCCA security, it is sufficient that an adversary is able to homomorphically evaluate the challenge ciphertext without this being detectable. Recall that a ciphertext from the Paillier encryption scheme has the following form $c = g^m \cdot r^n \mod n^2$. A Paillier ciphertext can be re-randomized by multiplying it by r'^n for some $r' \in \mathbb{N}$. In other words $c' = c \cdot r'^n \mod n^2 = g^{m_b} \cdot (r \cdot r')^n \mod n^2$ is another ciphertext encrypting m. Notice that $(r \cdot r') \mod n^2$ is uniformly distributed, therefore c' has the same distribution as c. It follows that for any polynomial-time algorithm $\mathsf{Extract}$, and any ciphertext pair (c, c') where c' is obtained by re-randomizing c,

$$\Pr[c \leftarrow \mathsf{Extract}(c')] \leq \mathsf{negl}(\lambda) \tag{7}$$

An adversary $\mathcal{A}$ against the IND-vCCA security of $\mathcal{E}'$, upon receiving the challenge ciphertext c^*, can re-randomize it as described above and query to its decryption oracle. It follows from Equation 7 that the oracle will send the decryption with overwhelming probability and $\mathcal{A}$ wins the IND-vCCA game. $\square$

A.3 Proof of Proposition 3

Proof. Akavia et al. [1] show that a sanitized version of a homomorphic encryption scheme, say FHEW [31] is FuncCPA. Now FHEW uses bootstrapping, i.e. an encryption of the secret key is released. It follows that sanitized FHEW cannot be IND-vCCA as an adversary can always query the bootstrapping key to trivially win the game.

A.4 Proof of Proposition 5

We recall the definition of IND-RCCA from [19].

Definition 12 (IND-RCCA **[19]). ***An encryption scheme* $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be* IND-RCCA *secure, if for any* PPT *adversary* $\mathcal{A}$*, it has negligible advantage in the indistinguishability game from Definition 2, with the following oracles.*

- $\mathcal{O}_1(c)$*: Return* $\mathsf{Dec}_{\mathsf{sk}}(c)$
- $\mathcal{O}_2(c)$*: Compute* $m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$*. If* $m \in \{m_0, m_1\}$ *return a predefined value Test. Otherwise, return* m*.*

Proof. Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be an IND-RCCA secure homomorphic encryption scheme. We show that it is also IND-vCCA secure. It follows from the IND-RCCA security of $\mathcal{E}$, that Eval can only re-randomize ciphertexts. Indeed, any further homomorphism allows to trivially win the IND-RCCA game. It follows that we can simply build an extraction algorithm using the $\mathcal{O}_2$ oracle. More precisely, let $\mathcal{A}$ be an adversary against the IND-vCCA security of $\mathcal{E}$. We build an adversary $\mathcal{B}$ against the IND-RCCA notion.

Upon receiving the public-key pk, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pk})$ and answers decryption queries as follows. In the first phase, it just uses $\mathcal{O}_1$. In the second phase, the extraction algorithm simply queries the $\mathcal{O}_2$ oracle and returns (Id, c^*) if the answer is Test and the symbol for fresh ciphertext otherwise. Hence, $\mathcal{B}$ uses $\mathcal{O}_2$ to obtain either m or the predetermined value Test. It then either returns m or the special symbol $\perp$ depending on the case.

When $\mathcal{A}$ outputs a guess b, $\mathcal{B}$ does the same. The soundness of the extraction algorithm is clear, therefore $\mathcal{B}$ perfectly simulates the IND-vCCA game for $\mathcal{A}$. The theorem follows. $\square$