What Have SNARGs Ever Done for FHE?

Michael Walter
michael.walter@zama.ai
Zama, France

Abstract

In recent years, there have been several constructions combining FHE with SNARGs to add integrity guarantees to FHE schemes. Most of these works focused on improving efficiency, while the precise security model with regards to client side input privacy has remained understudied. Only recently it was shown by Manulis and Nguyen (Eurocrypt'24) that this combination does not yield IND-CCA1 security. So an interesting open question is: does the SNARG actually add any meaningful security to input privacy? We address this question in this note and give a security definition that meaningfully captures the security of the FHE plus SNARG construction.

1 Introduction

FHE has gained a lot of traction in recent years and is being deployed in practice. It promises to unlock a wide range of applications and the first one that comes to mind is secure outsourcing of computation. For this, consider a client that holds some data and it wishes to compute a function on that data. In many settings, the client is not able to do so on its own due to resource constraints. So the client may send the inputs to a server, which computes the function and returns the results. We will focus on this scenario in this note.

This setting immediately raises at least two security concerns. The first is input privacy: since the client sends the input to the server in the clear, the server will trivially learn these inputs. In many applications, this data is sensitive and thus this might rule out such applications or introduce strong assumptions about the honesty of the server. FHE may be used to address this problem, as the client can encrypt the inputs and the server can still compute the function on the encrypted data and return the result in encrypted form.

The second security issue is integrity: how does the client know that a function was indeed computed correctly? FHE does not provide any guarantees for integrity. Note that this is also related to the privacy issue above as most practical FHE schemes [DM15, CGGI20, BGV12, CKKS17] are only passively secure[1],

[1] In fact, the scheme in [CKKS17] is not even *passively secure* for a natural definition of passive, as shown in [LM21].

meaning that they only protect input privacy for honest-but-curious servers, i.e. servers that do not deviate from the protocol. If the server is actively malicious, e.g. executing the incorrect function or tampering with ciphertexts, privacy of the client might be lost entirely [CGG16]. There are several approaches to add integrity to the system, see for example [VKH23] and references therein. The approach relevant to this work is a line of research that uses SNARGs [FNP20, BCFK21, GNS23, VKH23, ACGSV23, ABPS24, TTW24]. With the exception of [ACGSV23], the basic idea is to use a SNARG to prove the correct execution of the homomorphic evaluation of a function.

Adding this integrity check to the FHE system seems to also address the privacy concern with regards to malicious servers since it seems difficult now to deviate from the protocol. Indeed, in [VKH23] it was claimed that such a system is IND-CCA1 secure. Unfortunately, as shown in [MN24], this is incorrect. The reason is that an adversary may still tamper with the input ciphertexts and return a faulty ciphertext that was correctly computed using the homomorphic evaluation.

The work of [MN24] then showed how to construct IND-CCA1 secure schemes (and even stronger ones) by adding additional checks to the input ciphertexts. Of course, this comes with additional cost and it still leaves open the question: what security exactly is achieved by combining a SNARG with IND-CPA secure FHE? As it currently stands, the strongest known definition achieved by this construction is IND-CPA security - the same as the FHE scheme by itself. So one may ask if adding a SNARG (which is typically very expensive) actually provides any additional security guarantees. Previous work has typically focused on improving the practical efficiency of this approach, while the security model with regards to privacy has remained understudied.

In this work, we give a security definition, *semi-active* security (IND-SA), that exactly captures the privacy of the client data of such a construction and allows to deduce precisely, for which applications this construction is suitable and guarantees privacy of the client input. The definition is inspired by the observation that in our above application, the client first computes the input ciphertexts and "knows" the function it wants to be computed. So as long as it stores these, it may check the output ciphertext received from the server against these inputs. Since this requires full control over the encryptions by the client, it is natural to focus on the case of symmetric FHE, which we do here. We also show that IND-SA lies strictly in between IND-CPA and IND-CCA2 security (in fact, it is strictly in between IND-CPA and IND-vCCA) and is incomparable to IND-CCA1 (and also FuncCPA, IND-CCVA and IND-CCA1.5).

1.1 Open Problems

In this work we focus on symmetric FHE since we need full control of the inputs and we can leverage the fact that the client has state, which we assume cannot be tampered with. There are ways to obtain such control even in the public key setting using additional machinery like proofs of plaintext and state in a blockchain. But from a definitional standpoint, this is harder to capture

precisely, so we consider it out of scope for this work. We do believe that an extension of our definition to the public key case would be a valuable contribution.

Furthermore, in this work we focus on exact schemes. Embedding the definition in the landscape of security definitions for approximate schemes ([LM21, CFP+24]) is left to future work.

1.2 Related Work

We already mentioned the work of [MN24] that shows how to achieve an even stronger security notion than ours at the expense of adding additional checks on the well-formedness of the input ciphertext. While this is applicable in a wider range of contexts, it comes at the price of an added cost in the constructions. Note that the cost comprises not only proving and verifying that a ciphertext is wellformed. But the verification also has to be proven using the SNARK, at least in the fully compact case, which is likely to add significant overhead in practice. Furthermore, the constructions in [MN24] require SNARKs that are black-box, straightline, simulation extractable, which most practical SNARKs are not. In contrast, our definition is achievable using constructions merely requiring SNARGs. On the other hand, it is only applicable in the private key setting, where the user has full control over the ciphertexts, knows the function to be executed, and is able to store that information. So compared to [MN24], our definition yields a different trade-off in ease of achievability versus applicability. Finally, we remark that the definition of [MN24] is based on an *extractor*, which is arguably counter-intuitive in the context of indistunguishability notions for encryption schemes and yields a significantly more complex definition than ours.

We also mention that the idea of checking inputs and correct evaluation of input ciphertexts to achieve security against actively malicious adversaries actually predates the work of [MN24]. This approach was already proposed in [Sma23] using multiple evaluators to achieve integrity by consensus. The work of [Sma23] views FHE applications through an MPC lens and proves security for its construction using the UC model. The latter is simulation-based and thus stronger than game-based definitions, but for simplicity we focus on game-based definitions in this note.

Acknowledgment An earlier draft of this note incorrectly claimed that IND-SA implies FuncCPA. We are grateful to Jérôme Nguyen for pointing out that that is incorrect and the two notions are in fact incomparable.

2 Preliminaries

Compared to the typical definition, we extend the syntax of FHE and allow the decryption function to take additional inputs specifying the function and input ciphertexts that were used to obtain the ciphertext.

Definition 1. *Let $\mathcal{P}$, $\mathcal{C}$ and $\mathcal{F}$ be the plaintext space, ciphertext space and a function family, respectively. A symmetric FHE scheme $\mathcal{E}$ for $\mathcal{F}$ is a tuple of algorithms:*

- $\mathcal{E}.\mathrm{Gen}(1^\lambda)$*: generates a secret key s and an evaluation key p*
- $\mathcal{E}.\mathrm{Enc}(s, m)$*: takes a key s and message $m \in \mathcal{P}$ and outputs a ciphertext $c \in \mathcal{C}$*
- $\mathcal{E}.\mathrm{Dec}(s, c, f, (c_1, \ldots, c_\ell))$*: takes a secret key, a ciphertext $c \in \mathcal{C}$, a function in $\mathcal{F}$ and a tuple of input ciphertexts $(c_1, \ldots, c_\ell) \in \mathcal{C}^\ell$ and returns a message in $\mathcal{P}$.*
- $\mathcal{E}.\mathrm{Eval}(p, f, (c_1, \ldots, c_\ell))$*: takes an evaluation key p, a function in $\mathcal{F}$ and a tuple of input ciphertexts $(c_1, \ldots, c_\ell) \in \mathcal{C}^\ell$ and returns a ciphertext $\mathcal{C}$.*

In the following we will assume that $\mathcal{E}.\mathrm{Dec}$ and $\mathcal{E}.\mathrm{Eval}$ are deterministic. As stated above, there is a trivial construction of FHE from any encryption scheme simply by having $\mathcal{E}.\mathrm{Dec}$ decrypt all ciphertexts c_i and apply f to obtain the message. This construction is typically ruled out by requiring that the ciphertexts are *compact*, i.e. independent of the function f and input ciphertexts $c_1, \ldots, c_\ell$. As we make the function and input ciphertexts explicit inputs to the decryption function, this is not possible in our case. We could take the approach of [MN24] and not provide f as an input to the decryption. Looking ahead that would mean that the decryption algorithm does not check that f evaluated on $c_1, \ldots, c_\ell$ to obtain the ciphertext c, but rather that *some* $f' \in \mathcal{F}$ was used to obtain c. However, we believe that in applications that we have in mind it is actually very useful to be able check that a specific f was evaluated. There are still non-trivial contructions that leverage preprocessing of f and state to yield compact (or at least relaxed compact in the sense of [MN24]) decryption functions. In any case, we are more interested in the security of such constructions, so we ignore the compactness requirement.

An FHE scheme $\mathcal{E}$ is *correct* if for all $(m_1, \ldots, m_\ell) \in \mathcal{P}^\ell$ and all $f \in \mathcal{F}$ we have

$$\Pr\left[m \neq f(m_1, \ldots, m_\ell) \;\middle|\; \begin{array}{c} (s,p) \leftarrow \mathcal{E}.\mathrm{Gen}(1^\lambda) \\ c_i \leftarrow \mathcal{E}.\mathrm{Enc}(s, m_i) \text{ for all } i \\ c \leftarrow \mathcal{E}.\mathrm{Eval}(p, f, c_1, \ldots, c_\ell) \\ m \leftarrow \mathcal{E}.\mathrm{Dec}(s, c, f, c_1, \ldots, c_\ell) \end{array} \right] = \mathrm{negl}(\lambda) \ .$$

Definition 2. *A succinct non-interactive argument (SNARG) $\Pi_{SNARG} = (\mathrm{Gen}, \mathrm{Prove}, \mathrm{Verify})$ for a relation $\mathcal{R}$ is a triple of algorithms such that:*

1. Gen *takes as input a security parameter $\lambda \in \mathbb{N}$ and outputs a common reference string* crs.
2. Prove *takes as input* crs*, a statement x and a witness w and outputs a proof π when $(x, w) \in \mathcal{R}$.*

3. Verify *takes as input* crs*, a statement* x *and a proof* π *and outputs* Acc *or* Rej.

A SNARG is *complete* if for all $(x, w) \in \mathcal{R}$,

$$\Pr\left[\text{Verify}(\mathsf{crs}, x, \pi) = \text{Acc} \;\middle|\; \begin{array}{c} \mathsf{crs} \leftarrow \text{Gen}(\lambda) \\ \pi \leftarrow \text{Prove}(\mathsf{crs}, x, w) \end{array}\right] = 1 \ .$$

A SNARG is *sound* if for all PPT adversaries $\mathcal{A}$

$$\Pr\left[\begin{array}{c} \text{Verify}(\mathsf{crs}, x, \pi) = Acc \\ \wedge x \notin \mathcal{R} \end{array} \;\middle|\; \begin{array}{c} \mathsf{crs} \leftarrow \text{Gen}(\lambda) \\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] = \text{negl}(\lambda) \ .$$

2.1 FHE Security Notions

In this note we propose a new security definition and relate it to other notions from the literature. We use the multi-challenge (or Left-or-Right) version for all of our definitions. At times, we will still refer to the *challenge ciphertext*, in which case we simply mean the first encryption query where $m_0 \neq m_1$, i.e. the point at which the "left" and the "right" worlds diverge.

Definition 3. *A symmetric FHE scheme* $\mathcal{E}$ *is IND-CPA secure if for all PPT adversaries* $\mathcal{A}$ *it holds that*

$$\left|2 \cdot \Pr\left[b = b' \;\middle|\; \begin{array}{c} s \leftarrow \mathcal{E}.KeyGen() \\ b \leftarrow U(\{0,1\}) \\ b' \leftarrow \mathcal{A}^{O^b_{Enc_s}()} \end{array}\right] - 1\right| = \text{negl}(\lambda)$$

where $O^b_{Enc_s}(m_0, m_1)$ *is the oracle from Algorithm 1.*

Algorithm 1: Encryption Oracle $O^b_{Enc_s}(m_0, m_1; i)$

$c \leftarrow \mathcal{E}.Enc_s(m_b)$
$S[i] \leftarrow (m_0, m_1, c)$
$i \leftarrow i + 1$
return c, i

The other notions used in this work are extensions of the IND-CPA definition with varying versions of decryption oracles. Note that we adjust decryption oracles to match our modified syntax. This might be slightly confusing at first read, especially when other functions or ciphertexts are sent or extracted during the security game, so we invite the reader to ignore this modification on first read.

Definition 4. *A symmetric FHE scheme* $\mathcal{E}$ *is IND-CCA2 secure if all PPT adversaries* $\mathcal{A}$ *have negligible advantage in winning the IND-CPA game, where* $\mathcal{A}$ *additionally has access to the following decryption oracle:*

Algorithm 2: CCA Decryption Oracle $O_{Dec_s}(c, f, (c_1, \ldots, c_\ell))$

```
if ∃i : S[i] = (m_0, m_1, c) and m_0 ≠ m_1 then
    return ⊥
else
    return E.Dec_s(c, f, (c_1, ..., c_ℓ))
```

IND-CCA2 security is unachievable for homomorphic encryption schemes, but we may hope for IND-CCA1 security.

Definition 5. *A symmetric FHE scheme $\mathcal{E}$ is IND-CCA1 secure if all PPT adversaries $\mathcal{A}$ have negligible advantage in winning the IND-CPA game, where $\mathcal{A}$ additionally has oracle access to the CCA decryption oracle (Algorithm 3) only until receiving the challenge ciphertext.*

Until recently, it was assumed that FHE schemes that use bootstrapping (involving public bootstrapping keys) cannot be IND-CCA1 secure. (This assumption was proven incorrect in [MN24]). Unfortunately, most efficient FHE schemes do employ bootstrapping, so other relaxations of IND-CCA2 were investigated for certain applications [AGHV22].

Definition 6. *A symmetric FHE scheme $\mathcal{E}$ is FuncCPA secure if all PPT adversaries $\mathcal{A}$ have negligible advantage in winning the IND-CPA game, where $\mathcal{A}$ additionally has access to a re-encryption oracle that takes a ciphertext c (along with f' and $(c_1, \ldots, c_\ell)$) and a function f and returns $\mathcal{E}.\mathrm{Enc}(f(\mathcal{E}.\mathrm{Dec}(c, f', (c_1, \ldots, c_\ell))))$.*

Interestingly, in [AGHV22] is was suggested that FuncCPA is incomparable to IND-CCA1. However, one may check that the proof of IND-vCCA $\Rightarrow$ FuncCPA in [MN24] applies also to IND-CCA1 secure schemes as the "post-challenge" decryption oracle is never used. Accordingly, we also have IND-CCA1 $\Rightarrow$ FuncCPA and since the FuncCPA constructions in [AGHV22] are not IND-CCA1 secure, FuncCPA is strictly weaker than IND-CCA1.

One may also weaken the decryption oracle to match certain attack scenarios in practice, which was essentially done for IND-CCVA security.

Definition 7. *A symmetric FHE scheme $\mathcal{E}$ is IND-CCVA secure if all PPT adversaries $\mathcal{A}$ have negligible advantage in winning the IND-CPA game, where $\mathcal{A}$ additionally has access to a ciphertext validation oracle that takes a ciphertext c along with f and $(c_1, \ldots, c_\ell)$ and returns $\perp$ if $\mathcal{E}.\mathrm{Dec}(c, f, (c_1, \ldots, c_\ell)) = \perp$ and $\top$ otherwise.*

One may combine IND-CCVA security and IND-CCA1 security to obtain a slightly stronger notion.

Definition 8. *A symmetric FHE scheme $\mathcal{E}$ is IND-CCA1.5 secure if all PPT adversaries $\mathcal{A}$ have negligible advantage in winning the IND-CCA1 game, where $\mathcal{A}$ additionally has access to the ciphertext validation oracle defined in Definition 7.*

Finally, we give the IND-vCCA definition. While originally proposed in [MN24], we will focus on the IND-vCCA definition from [CFP+24], because it is closer in style to our definition and thus a little easier to hightlight similarities and differences. Note that the two definitions from [MN24] and [CFP+24] were proven equivalent for exact FHE schemes in [CFP+24].

The IND-vCCA definition requires that a scheme $\mathcal{E}$ has two disjoint sets of ciphertexts, $\mathcal{C}_1$ and $\mathcal{C}_2$, where $\mathcal{C}_1$ is the range of $\mathcal{E}.\mathrm{Enc}$ and $\mathcal{C}_2$ is the range of $\mathcal{E}.\mathrm{Eval}$. The set $\mathcal{C}_1$ is called *fresh ciphertexts*.

Definition 9. *For an FHE scheme $\mathcal{E}$ a* sound extractor $\mathcal{E}.\mathrm{Extract} : \mathcal{C} \mapsto \mathcal{F} \times \mathcal{C}_1^*$ *is defined as an algorithm that on input:*

- $c \in \mathcal{C}_1$ *returns* (Id, c)*, and*
- $c \in \mathcal{C}_2$ *returns* $(f \in \mathcal{F}, (c_1, \ldots, c_\ell) \in \mathcal{C}_1^\ell)$ *such that*

$$\mathcal{E}.\mathrm{Dec}(c) = f(\mathcal{E}.\mathrm{Dec}(c_1), \ldots, \mathcal{E}.\mathrm{Dec}(c_\ell)) \ .$$

With this definition in place, we can define IND-vCCA security in the following way. Note that we adopt the convention from [CFP+24] in that we assume $f(m_1, \ldots, m_\ell) = \perp$ if $m_i = \perp$ for any i.

Definition 10. *A symmetric FHE scheme $\mathcal{E}$ is IND-vCCA secure if all PPT adversaries $\mathcal{A}$ have negligible advantage in winning the IND-CPA game, where $\mathcal{A}$ additionally has access to the following decryption oracle:*

Algorithm 3: vCCA Decryption Oracle $O_{Dec_s}(c, f', (c'_1, \ldots, c'_\ell))$

```
(f, c_1, ..., c_ℓ) ← E.Extract(c)
for j ∈ 1, ..., ℓ do
    if ∃i : S[i].c = c_i then
        (m'_j, m''_j) ← (S[i].m_0, S[i].m_1)
    else
        (m'_j, m''_j) ← (⊥, ⊥)
end
if f(m'_1, ..., m'_ℓ) ≠ f(m''_1, ..., m''_ℓ) then
    return ⊥
else
    return E.Dec_s(c, f', (c'_1, ..., c'_ℓ))
```

3 Semi-Active Security for Symmetric FHE

Consider a query-based application, where the client encrypts its input and sends it to the sever for evaluation, which in turn responds with the result of the computation. A natural approach to define active security would be to reflect

the query-based interaction in the security game, where the challenger sends queries in the form of functions and ciphertexts and the adversary responds with ciphertexts. The latter are decrypted and returned to the adversary. If the challenger never sends the challenge ciphertext in the form of a query, we obtain a security notion similar to [MN24], but where active security is much easier to achieve. Intuitively, the queries tie the input ciphertexts to the output ciphertexts and fulfil the role of the extractor from [MN24]. Then, combining a passively secure FHE scheme with a SNARK to verify correct evaluation should be sufficient.

There is a slight definitional issue in that it is unclear which distributions the messages and queries should be drawn from. We use the typical cryptographic approach in letting the adversary pick them. This is straight-forward in the symmetric key setting, since the adversary needs to query an encryption oracle to obtain ciphertexts of known messages and thus we arrive at the following security definition.

Definition 11 (IND-SA security)**.** *A symmetric FHE scheme $\mathcal{E}$ is semi-actively secure if for all PPT adversaries $\mathcal{A}$ it holds that*

$$\left| 2 \cdot \Pr\left[b = b' \;\middle|\; \begin{array}{c} (s,p) \leftarrow \mathcal{E}.KeyGen() \\ b \leftarrow U(\{0,1\}) \\ b' \leftarrow \mathcal{A}^{O_{Enc_s}, O_{Dec_s}}(p) \end{array} \right] - 1 \right|$$

is negligible, where O_{Enc_s} and O_{Dec_s} are defined as in Algorithm 1 and Algorithm 4, respectively.

Algorithm 4: Decryption Oracle $O_{Dec_s}(c, f, (i_1, \ldots, i_\ell))$

if $\exists i_j \notin S$ **then**
  return $\perp$
$m_0 \leftarrow f(S[i_1].m_0, \ldots, S[i_\ell].m_0)$
$m_1 \leftarrow f(S[i_1].m_1, \ldots, S[i_\ell].m_1)$
if $m_0 = m_1$ **then**
  return $\mathcal{E}.Dec_s(c, f, (S[i_1].c, \ldots, S[i_\ell].c))$
else
  return $\perp$

This definition is heavily inspired by the IND-CPA^D from [LM21], but strengthened to allow the adversary to submit any ciphertext to the decryption oracle. The caveat is that the adversary must specify, how it claims the ciphertext was computed. This information is forwarded to the decryption algorithm that may now use this information to verify this claim. This is to reflect the fact that in the application we assume that the client sent a specific query to the server that it expects to be evaluated.

4 Relations to Other Notions

Notation In this section we will use the notation Left $\Rightarrow$ Right and Left $\not\Rightarrow$ Right for two security definitions Left and Right. By the former, we mean that the Left definition implies the Right definition. By the latter, we mean that there is a separation between the two definitions in the sense that there exists a scheme that satifies the Left definition but not the Right definition.

We first show that IND-SA lies strictly in between IND-vCCA and IND-CPA.

Lemma 1. *IND-vCCA $\Rightarrow$ IND-SA $\Rightarrow$ IND-CPA and IND-CPA $\not\Rightarrow$ IND-SA $\not\Rightarrow$ IND-vCCA.*

Proof. IND-vCCA $\Rightarrow$ IND-SA: Let $\mathcal{E}$ be an IND-vCCA secure FHE scheme. It is not IND-SA secure as is, since the adversary may lie about the input ciphertexts and function it used to compute a ciphertext submitted to the decryption oracle and thus bypass the equality check in the IND-SA decryption oracle. However, a natural modification of $\mathcal{E}$ does yield an IND-SA secure scheme: Recall that the IND-vCCA security requires the existence of an *extractor* that can recover the function and input ciphertexts from a ciphertext derived through $\mathcal{E}.\mathrm{Eval}$. So $\mathcal{E}'.\mathrm{Dec}$ first calls the extractor and checks that the recovered input ciphertexts and function matches the ones claimed by the adversary in the IND-SA security game. If not, it returns $\perp$, otherwise it calls $\mathcal{E}.\mathrm{Dec}$ and returns the result. The implication follows from the observation that the two games are now syntactically the same.

IND-SA $\Rightarrow$ IND-CPA: This follows simply from the fact that the IND-CPA game is the same as the IND-SA game without decryption oracle.

IND-CPA $\not\Rightarrow$ IND-SA: None of the basic IND-CPA secure FHE schemes from the literature are IND-SA secure.

IND-SA $\not\Rightarrow$ IND-vCCA: Below we show that IND-SA $\not\Rightarrow$ IND-CCA1, so the claim follows from IND-vCCA $\Rightarrow$ IND-CCA1 [MN24]. □

We now establish through a series of lemmas that IND-SA is incomparable to IND-CCA1, IND-CCVA, and IND-CCA1.5.

Lemma 2. *IND-CCA1.5 $\not\Rightarrow$ IND-SA.*

Proof. We construct an IND-CCA1.5 secure FHE scheme $\mathcal{E}'$ from an IND-vCCA secure FHE scheme $\mathcal{E}$ and show that it is not IND-SA secure. For simplicity, restrict the set of functions to the ones with two inputs, i.e. $\mathcal{F} = \{\mathcal{P}^2 \mapsto \mathcal{P}\}$. We assume that $\mathcal{E}$ is also IND-SA secure by undergoing the transformation presented in the proof of Lemma 1.

We leave $\mathcal{E}.\mathrm{Gen}$, $\mathcal{E}.\mathrm{Enc}$ and $\mathcal{E}.\mathrm{Eval}$ unchanged, but modify $\mathcal{E}.\mathrm{Dec}$ such that $\mathcal{E}'.\mathrm{Dec}(c, f, c_1, c_2)$ first calls $m \leftarrow \mathcal{E}.\mathrm{Dec}(c, f, c_1, c_2)$. If $m \neq \perp$, return m. Otherwise, compute $m' \leftarrow \mathcal{E}.\mathrm{Dec}(c, f, c_2, c_1)$ and return m'.

We first argue that $\mathcal{E}'$ is IND-CCA1.5 secure. Assume it is not. Then an adversary against the IND-CCA1.5 security of $\mathcal{E}$ can faithfully simulate the decryption and ciphertext verification oracle to an IND-CCA1.5 adversary against

$\mathcal{E}'$ by leveraging its own decryption and ciphertext verification oracle and mimicking the decryption procedure of $\mathcal{E}'$. The claim follows from the fact that IND-vCCA $\Rightarrow$ IND-CCA1.5 [MN24].

We now show that $\mathcal{E}'$ is not IND-SA secure. Let $\mathcal{P} = \{0, 1\}$ and $f : \mathcal{P}^2 \mapsto \mathcal{P}$ be such that $f(x_0, x_1) = (x_0 \vee 1) \wedge x_1$. Note that f depends on the second input but not on the first. Now query the encryption oracle on

- $(0, 0)$ and receive $c' \leftarrow \mathcal{E}'.\mathrm{Enc}(s, 0)$ and index i' and
- $(m_0, m_1) = (0, 1)$ and receive $c^* \leftarrow \mathcal{E}'.\mathrm{Enc}(s, m_b)$ and index i^*.

Now compute $c \leftarrow \mathcal{E}'.\mathrm{Eval}(f, c', c^*)$ and query the decryption oracle on $(c, f, (i^*, i))$. Since $f(m_0, 0) = f(m_1, 0) = 0$, this is a valid IND-SA decryption query and thus $\mathcal{E}'.\mathrm{Dec}(c, f, c^*, c')$ will be called. Note that $\mathcal{E}.\mathrm{Dec}(c, f, c^*, c')$ will return $\perp$ due to the additional check in the IND-vCCA-to-IND-SA security transformation, due to the incorrect ordering of the input ciphertexts. Accordingly, $\mathcal{E}'.\mathrm{Dec}$ will return $\mathcal{E}.\mathrm{Dec}(c, f, c', c^*) = m_b$ and thus allow the adversary to win the IND-SA game. □

Lemma 3. *IND-SA $\not\Rightarrow$ IND-CCA1.*

Proof. Note that in Section 5 we present a construction that is IND-SA secure. The construction is essentially the one from [VKH23] that was shown not to be IND-CCA1 secure in [MN24]. □

Lemma 4. *IND-SA $\not\Rightarrow$ IND-CCVA.*

Proof. Let $\mathcal{E}$ an IND-CPA secure FHE scheme that is vulnerable to an IND-CCVA attack. Apply the construction of Section 5 to obtain an IND-SA secure scheme $\mathcal{E}'$. Note that $\mathcal{E}.\mathrm{Dec}$ can be simulated in the new scheme by evaluating the identity function on the ciphertext and submitting the result to $\mathcal{E}'.\mathrm{Dec}$. The same holds for the ciphertext verification oracle. Thus, any IND-CCVA attack on $\mathcal{E}$ will still be applicable to $\mathcal{E}'$. □

Summarizing the results from the three lemmas, we have

- IND-SA $\not\Rightarrow$ IND-CCA1 and IND-CCA1 $\not\Rightarrow$ IND-SA,
- IND-SA $\not\Rightarrow$ IND-CCVA and IND-CCVA $\not\Rightarrow$ IND-SA,
- IND-SA $\not\Rightarrow$ IND-CCA1.5 and IND-CCA1.5 $\not\Rightarrow$ IND-SA.

Finally, we show that IND-SA is strictly stronger than FuncCPA.

Lemma 5. *FuncCPA $\not\Rightarrow$ IND-SA and IND-SA $\not\Rightarrow$ FuncCPA.*

This proof is similar to the equivalent proofs for IND-vCCA (see [MN24], Theorem 4 and Proposition 3), so we only sketch the proofs.

Proof. IND-SA $\not\Rightarrow$ FuncCPA: Take an arbitrary IND-SA secure scheme and apply the transformation from [AGHV22] (Theorem 10). This does not affect IND-SA security, but the theorem shows that the resulting scheme is not FuncCPA secure.[2]

FuncCPA $\not\Rightarrow$ IND-SA: This follows by a similar argument as in [MN24]. The work of [AGHV22] shows that sanitized HE schemes are FuncCPA secure, while they do not have a machanism to check correct evaluation. Accordingly, they cannot be IND-SA secure, since the decryption oracle can trivially be used to win the game. □

5 FHE plus SNARK is IND-SA Secure

Let $\mathcal{E}$ be a IND-CPA secure FHE scheme for function family $\mathcal{F}$ with ciphertext space $\mathcal{C}$ and let Π be a SNARG for the relation

$$\mathcal{R} = \{(p, c, f, c_1, \ldots, c_\ell) \mid c = \mathcal{E}.\mathrm{Eval}(p, f, c_1, \ldots, c_\ell)\} \ .$$

The following construction of an FHE scheme $\mathcal{E}'$ is common in the literature.

- $\mathcal{E}'.\mathrm{Gen}(1^\lambda)$ computes $(s, p) \leftarrow \mathcal{E}.\mathrm{Gen}(1^\lambda$ and $\mathsf{crs} \leftarrow \Pi.\mathrm{Gen}(1^\lambda)$ and returns secret key (s, p, crs) and evaluation key (p, crs)
- $\mathcal{E}'.\mathrm{Enc}((s, p, \mathsf{crs}), m)$ computes $c \leftarrow \mathcal{E}.\mathrm{Enc}(s, m)$ and returns c
- $\mathcal{E}'.\mathrm{Eval}((p, \mathsf{crs}), f, c_1, \ldots, c_\ell)$ computes $c \leftarrow \mathcal{E}.\mathrm{Eval}(p, f, c_1, \ldots, c_\ell)$ and $\pi \leftarrow \Pi.\mathrm{Prove}(\mathsf{crs}, ((p, c, f, c_1, \ldots, c_\ell)))$ and returns (c, π)
- $\mathcal{E}'.\mathrm{Dec}((s, p, \mathsf{crs}), (c, \pi), f, c_1, \ldots, c_\ell)$ computes

$$d = \Pi.\mathrm{Verify}(\mathsf{crs}, ((p, c, f, c_1, \ldots, c_\ell)), \pi) \tag{1}$$

  and returns $\mathcal{E}.\mathrm{Dec}(s, c)$ if $d = \mathrm{Acc}$ and $\perp$ otherwise.

Lemma 6. *If $\mathcal{E}$ is IND-CPA secure and Π is sound, then $\mathcal{E}'$ is IND-SA secure.*

Proof. Let $\mathcal{A}$ be an adversary against the IND-SA security of $\mathcal{E}'$. Then it is easy to build an adversary $\mathcal{B}$ against the IND-CPA security of $\mathcal{E}$ assuming soundness of Π: the encryption oracle (Algorithm 1) is simulated by using the IND-CPA encryption oracle in Line 1. The decryption oracle (Algorithm 4) is simulated by performing the check in (Equation (1)) and returning m_0 from Line 3 if it succeeds (assuming $m_0 = m_1$). The correctness of this simulation follows from the correctness of $\mathcal{E}$ and the soundness of Π. □

[2]This observation is to Jérôme Nguyen.

5.1 Performance in the Outsourcing Application

Consider the query-based model, where a client wants to privately outsource some heavy computation by encrypting the inputs and asking the server to perform the computation.

As defined above the construction is asymptotically not any better for the client than performing the computation itself, since the decryption takes f and the $c_1, \ldots, c_\ell$ as input. Simply reading this input is essentially as expensive as performing the computation itself. However, the construction bears the potential of large asymptotic savings if using state and preprocessing. For example, recall that in our target applications the client sends f and $c_1, \ldots, c_\ell$ to the server to obtain the output. Clearly, any sane implementation will store these inputs so that they do not need to be sent back by the server.

Furthermore, by re-using the preprocessing of SNARGs like [GWC19, Gro16, BBB$^+$18, COS20] the verification of the proof and thus decryption can be very efficient. The preprocessing is linear in the circuit size of the function (but independent of the input) and produces some "compressed" information about the circuit for the verifier. Of course, it is not obvious which party should perform the preprocessing. If the client performs it itself, this might defeat the purpose of using FHE since a linear preprocessing is similarly expensive as performing the computation in the first place. Still, this approach can be useful if the same function is applied multiple times, such that the preprocessing effort of the client can be amortized over many computations. An example could be database or ML queries. Another approach could be to let the server perform the preprocessing if this preprocessing is transparent (for example using [BBHR18]), but this begs the question how the client can check the preprocessing without spending too much computational effort. A plausible approach could require the server to commit to the output of the preprocessing by, e.g., uploading the verifier input (or the hash value thereof) to a blockchain. Since anyone can perform the preprocessing, anyone can also re-run it and check the server's honesty. So a cheating server runs a very high risk of being caught by someone (not necessarily a client, but maybe a competitor).

References

[ABPS24] Shahla Atapoor, Karim Baghery, Hilder V. L. Pereira, and Jannik Spiessens. Verifiable FHE via lattice-based SNARKs. *IACR Communications in Cryptology*, 1(1), 2024. doi:10.62056/a6ksdkp10.

[ACGSV23] Diego F. Aranha, Anamaria Costache, Antonio Guimarães, and Eduardo Soria-Vazquez. HELIOPOLIS: Verifiable computation over homomorphically encrypted data from interactive oracle proofs is practical. Cryptology ePrint Archive, Paper 2023/1949, 2023. https://eprint.iacr.org/2023/1949. URL: https://eprint.iacr.org/2023/1949.

[AGHV22] Adi Akavia, Craig Gentry, Shai Halevi, and Margarita Vald. Achievable CCA2 relaxation for homomorphic encryption. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part II*, volume 13748 of *LNCS*, pages 70–99. Springer, Heidelberg, November 2022. doi:10.1007/978-3-031-22365-5_3.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018. doi:10.1109/SP.2018.00020.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018. doi:10.4230/LIPIcs.ICALP.2018.14.

[BCFK21] Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. Flexible and efficient verifiable computation on encrypted data. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 528–558. Springer, Heidelberg, May 2021. doi:10.1007/978-3-030-75248-4_19.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012. doi:10.1145/2090236.2090262.

[CFP+24] Sébastien Canard, Caroline Fontaine, Duong Hieu Phan, David Pointcheval, Marc Renard, and Renaud Sirdey. Relations among new CCA security notions for approximate FHE. Cryptology ePrint Archive, Paper 2024/812, 2024. https://eprint.iacr.org/2024/812. URL: https://eprint.iacr.org/2024/812.

[CGG16] Ilaria Chillotti, Nicolas Gama, and Louis Goubin. Attacking FHE-based applications by software fault injections. Cryptology ePrint Archive, Report 2016/1164, 2016. https://eprint.iacr.org/2016/1164.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, January 2020. doi:10.1007/s00145-019-09319-x.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages

409–437. Springer, Heidelberg, December 2017. doi:10.1007/978-3-319-70694-8_15.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 769–793. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45721-1_27.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46800-5_24.

[FNP20] Dario Fiore, Anca Nitulescu, and David Pointcheval. Boosting verifiable computation on encrypted data. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 124–154. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45388-6_5.

[GNS23] Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: SNARKs for ring arithmetic. *Journal of Cryptology*, 36(4):41, October 2023. doi:10.1007/s00145-023-09481-3.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_11.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.

[LM21] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 648–677. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77870-5_23.

[MN24] Mark Manulis and Jérôme Nguyen. Fully homomorphic encryption beyond IND-CCA1 security: Integrity through verifiability. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 63–93. Springer, Heidelberg, May 2024. doi:10.1007/978-3-031-58723-8_3.

[Sma23] Nigel P. Smart. Practical and efficient FHE-based MPC. In *IMACC*, volume 14421 of *Lecture Notes in Computer Science*, pages 263–283. Springer, 2023.

[TTW24] Louis Tremblay Thibault and Michael Walter. Towards verifiable FHE in practice: Proving correct execution of TFHE's bootstrapping using plonky2. Cryptology ePrint Archive, Paper 2024/451, 2024.

[VKH23] Alexander Viand, Christian Knabenhans, and Anwar Hithnawi. Verifiable fully homomorphic encryption, 2023. `arXiv:2301.07041`.