Security with Functional Re-Encryption from CPA

Yevgeniy Dodis[*]
New York University

Shai Halevi
AWS

Daniel Wichs[†]
Northeastern and NTT Research

May 2023

Abstract

The notion of *functional re-encryption security (funcCPA)* for public-key encryption schemes was recently introduced by Akavia *et al.* (TCC'22), in the context of homomorphic encryption. This notion lies in between CPA security and CCA security: we give the attacker a *functional re-encryption oracle* instead of the decryption oracle of CCA security. This oracle takes a ciphertext ct and a function f, and returns fresh encryption of the output of f applied to the decryption of ct; in symbols, $\mathsf{ct}' = \mathrm{Enc}(f(\mathrm{Dec}(\mathsf{ct})))$. More generally, we even allow for a multi-input version, where the oracle takes an arbitrary number of ciphetexts $\mathsf{ct}_1, \ldots \mathsf{ct}_\ell$ and outputs $\mathsf{ct}' = \mathrm{Enc}(f(\mathrm{Dec}(\mathsf{ct}_1), \ldots, \mathrm{Dec}(\mathsf{ct}_\ell)))$.

In this work we observe that funcCPA security may have applications beyond homomorphic encryption, and set out to study its properties. As our main contribution, we prove that funcCPA is "closer to CPA than to CCA"; that is, funcCPA secure encryption can be constructed in a black-box manner from CPA-secure encryption. We stress that, prior to our work, this was not known even for basic re-encryption queries corresponding to the identity function f.

At the core of our result is a new technique, showing how to handle *adaptive* functional re-encryption queries using tools previously developed in the context of non-malleable encryption, which roughly corresponds to a single *non-adaptive* parallel decryption query.

1 Introduction

The notion of functional re-encryption *FuncCPA* security for encryption schemes was recently introduced by Akavia *et al.* [AGHV22], and shown to be useful in the context of homomorphic encryption schemes. This notion is similar to CCA security, except that the attacker is given a *re-encryption oracle* rather than a decryption oracle. Roughly, the oracle replies to a query ciphertext ct with another ciphertext $\mathsf{ct}' = \mathrm{Enc}(\mathrm{Dec}(\mathsf{ct}))$, corresponding to a fresh encryption of the message contained in ct. More generally, the definition even permits *"functional" re-encryption queries*: the attacker also specifies a function f, and the oracle returns $\mathsf{ct}' = \mathrm{Enc}(f(\mathrm{Dec}(\mathsf{ct})))$. Or even more generally, we can consider *"multi-input functional" re-encryption queries*, where the oracle takes an arbitrary number of ciphetexts $\mathsf{ct}_1, \ldots \mathsf{ct}_\ell$ and outputs $\mathsf{ct}' = \mathrm{Enc}(f(\mathrm{Dec}(\mathsf{ct}_1), \ldots, \mathrm{Dec}(\mathsf{ct}_\ell)))$. Below, when we say FuncCPA, we refer to the strongest notion with multi-input functional queries by default, unless we explicitly restrict to single-input functional re-encryption or non-functional re-encryption.

At first glance, the FuncCPA-oracle may seem quite useless to the attacker, as it only returns properly encrypted ciphertexts. One may even be tempted to assume that every CPA-secure scheme is also FuncCPA-secure. Surprisingly, this is not the case: Akavia *et al.* described in [AGHV22] a CPA-secure scheme where a single (non-functional) re-encryption query allows the adversary to recover the secret key. This example makes FuncCPA an interesting notion to study, as it lies "somewhere in between" CPA and CCA security.

[*]Research Supported by NSF grant CNS-2055578, and gifts from JP Morgan, Protocol Labs and Algorand Foundation.
[†]Research supported by NSF grant CNS-1750795, CNS-2055510 and the JP Morgan faculty research award.

FuncCPA for Non-Homomorphic Schemes. Although Akavia *et al.* only considered FuncCPA in the context of homomorphic encryption, we note that it makes sense also for schemes that are not homomorphic. For example, consider using a "secure enclave" (such as a secure hardware or trusted execution environment) to address the same client-server delegation scenario. In this setting there could be an "analyst" that wants to perform various studies on sensitive data, multiple clients who are willing to donate their data to those studies (as long as their privacy is respected), and a worker server on which the studies are computed, endowed by a secure enclave. The analyst will have a secret-public key pair, they will send the secret key over a secure channel to the enclave, and publish the corresponding public key. Clients who want to donate their data to the studies will encrypt it under the analyst's public key, and send the ciphertext to the server. The server will collect the data (possibly more and over over time), and occasionally will ask the enclave to compute something on the the encrypted data. The enclave will decrypt the given pieces of data, compute the required function, encrypt the result, and return to the server.[1] When each study is over, the server will send the end-result back to the analyst, to be decrypted and used.

In that setting, we note that the queries made by the server to its secure enclave are exactly the type of re-encryption queries that we consider: The server sends encrypted data and some function, and the enclave decrypts, computes the function, re-encrypts, and return to the server.

Our Main Question. In this work we set out to study the properties of FuncCPA security. For starters, we give a simple proof that every CCA-secure scheme is also FuncCPA -secure.[2] Having established that FuncCPA security is implied by CCA security and implies CPA security, the main question that we address in this work is whether FuncCPA is "more like" CPA or CCA. Specifically we ask:

Can one construct a FuncCPA-secure encryption scheme from any CPA-secure one?

We stress that the answer to this question is unknown even if we restrict to basic ("non-functional") re-encryption queries $\mathrm{Enc}(\mathrm{Dec}(\mathsf{ct}))$, corresponding to a single ciphertext with the identity function f.

The relation between CPA-secure schemes and CCA-secure ones was studied extensively in the literature, and many constructions of the latter are known. However, all these constructions either require making extra assumptions beyond just the existence of CPA-secure schemes [NY90, Sah99, MSS13, HKW20], or are carried out in idealized models (e.g., [FO99]). In particular, whether one can construct a CCA secure scheme generically from any CPA secure one, is considered a major open problem in cryptography.

As for standard model constructions from CPA encryption without extra assumptions, it is known that the existence of CPA-secure encryption can generically be upgraded to weaker variants of CCA security, such as non-malleability [PSV06, CDMW18a, CDMW18b], bounded CCA security [CHH$^+$07], or security against self-destruct attacks [CDM$^+$20]. Of particular interest to us, non-malleable encryption corresponds to a "non-adaptive" variant of CCA-security, where the adversary can only issue one set of non-adaptive decryption queries in parallel. Pass *et al.* [PSV06] showed how to generically transform CPA-security to non-malleability, and Choi *et al.* [CDMW18a, CDMW18b] showed that this can even be done while using the underlying CPA-secure scheme as a black box.

1.1 Our Main Result

We show that FuncCPA is "more like CPA than CCA", specifically we prove:

Theorem 1. *If CPA-secure encryption schemes exist, then so do FuncCPA-secure encryption schemes. Moreover, the transformation can be made black-box in the underlying CPA-secure scheme.*

[1] Notice, this application requires FuncCPA security for queries consisting of multiple ciphertexts, which is why this will be our default notion of FuncCPA security.

[2] In fact, we show (see Lemma 8) that FuncCPA security is implied by CCA security against "lunchtime attacks", known as CCA1.

Perhaps surprisingly, the transformation that we describe here is identical to the CPA-to-non-malleable transformation of Choi *et al.* from [CDMW18a], except that *we need to start with a scheme which is already non-malleable.* Therefore, one way to get FuncCPA from CPA in a black-box way is to apply the transformation from [CDMW18a] once to get non-malleability, then apply it again to get FuncCPA -security.[3]

Our Technique. For simplicity of notation, we describe our technique for the case of single-input functional re-encryption queries, but everything trivially generalizes to multi-input functional re-encryption queries as well. The main difficulty of a reduction from CPA to FuncCPA security is that we need to simulate adaptive queries to a functional re-encryption oracle, which involves decryption. How do we do this without a decryption oracle? We use the transformation from [CDMW18a], which was designed to allow a simulation of a batch of non-adaptive queries to a decryption oracle, and show that the same approach also allows us to simulate adaptive queries to a functional re-encryption oracle.

The high-level structure of the transformation from [CDMW18a] is to encode the message with an appropriate error-correcting code, then encrypt the resulting codeword symbols multiple times under different keys, and check on decryption that the decrypted words are close enough to each other. Thinking of the encrypted symbols as a matrix, with the rows corresponding to multiple encryptions and the columns corresponding to positions in the codeword, Choi *et al.* observed that checking closeness can be done just by verifying that the codewords agree on some small randomly chosen subset of the columns. Hence the decryptor only needs to know the secret keys for one row to do the actual decryption, and for that small subset of columns to do the checks. Security is then proven by reduction to the security of the keys for which the decryptor does not know the secret keys.

For our purposes, we consider a "bad event" in which the attacker submits a query ciphertext for which not all the rows are close to each other, but this "is not caught" by the checks on decryption. As long as this bad event does not happen, we can show that the decryption procedure will always give the same answer, no matter which row or columns are used in it. Hence, as long as this bad event does not happen, one can simulate the attacker's view without knowing too many keys. Moreover, we also show that as long as the bad event did not happen so far, the adversary does not have enough information to cause it to happen in the next query. This allows us to describe a reduction using "almost-functional" keys, where the reduction can decrypt all the attacker's queries *except the one that it will use for its challenge ciphertext.* This, in turn, lets us switch from $\mathrm{Enc}(f(\mathrm{Dec}(\mathsf{ct})))$ to $\mathrm{Enc}(0)$, one query at a time.

Importantly, to turn the advantage of the FuncCPA attacker into an advantage in attacking the underlying scheme, the reduction algorithms that we describe must know if the bad event occurred on *any* of the re-encrypted ciphertexts ct_i. The key novelty here is the observation that the reduction does not need to know this at the time of each (functional) re-encryption query. Instead, it can run the attacker until the end, and check if the bad event had occurred on any of the ciphertexts $(\mathsf{ct}_1, \ldots, \mathsf{ct}_q)$ only then. This check requires access to the decryption oracle, which is where we use the *non-malleability* of the underlying encryption scheme (rather than mere CPA security). Namely, non-malleability allows us to make one such parallel decryption query to know precisely when any bad event happened.[4] Hence, the reduction to non-malleability will use a single, *non-adaptive* parallel decryption query to check whether or not the bad event occurred in any of the attacker's queries, despite the attacker making many *adaptive* functional re-encryption queries. See more details in section 3.

We note that Choi *et al.* later described a more efficient CPA-to-non-malleability transformation [CDMW18b], and that the same line of reasoning probably works for that transformation as well. But since we do not focus on efficiency in this work, we use the (arguably simpler) transformation from [CDMW18a].

Do we Really Need Two Transformations? Seeing how we need to apply the same transformation twice, once to move from CPA to non-malleability and a second time to get FuncCPA, it is natural to ask if we can spare one of them – can't we just apply this transformation once?

[3]Our transformation in Theorem 1 is not only FuncCPA-secure, but also non-malleable. See Remark 1.

[4]This aspect was not needed in the analysis of [CDMW18a], as they did not have any re-encryption queries.

While we don't know the answer, our proof technique completely breaks down without the assumption that the underlying scheme is already non-malleable. In fact, our intuition is that CPA security of the underlying scheme is not enough to ensure FuncCPA of the result, and one should be able to exhibit a counter-example using strong enough homomorphic properties of it. We note, however, that such counter-example "cannot be too simple", since the transformation encrypts different codeword positions with different keys, meaning that any such example will be at least as hard as showing that bit-encryption does not imply FuncCPA. Still, we conjecture that non-malleability is really needed for our proof.

Variations of FuncCPA. As we mentioned above, there are different variants of FuncCPA -security, with one or more ciphertexts, and with or without functional queries. Akavia *et al.* have shown in [AGHV22] that these notion are all equivalent for homomorphic encryption schemes,[5] but generally they may differ.

In fact, for our purposes it is convenient to use a possibly-stronger formal definition than the one from [AGHV22]. (This stronger definition was also considered by Akavia and Vald [AV22].) Roughly, instead of only requiring that functional re-encryption queries cannot help the attacker break semantic security, we require that these queries are indistinguishable from fresh encryptions of some fixed message (e.g. 0). This clearly implies that such queries cannot help break semantic security, but the converse may not hold. We denote this potentially stronger notion by FuncCPA$^+$.

We note that, prior to the current work, constructing FuncCPA from CPA-secure encryption is challenging even for the weakest of these notions (non-functional re-encryption queries not helping break semantic security), while our positive result applies even to the strongest of them (functional re-encryption queries with multiple ciphertexts look like encryptions of 0). We discuss implications and separations between many of these variants in section 2.1 and in appendix A.

Organization

In section 2 we recall the basic definitions and prove some simple properties of FuncCPA-security. Our main result theorem 1 is proved in section 3, and we state a few open problems in section 4. Finally, in appendix A we prove some relations between various security notions.

2 Definitions

Signatures. A signature scheme $\mathcal{S} = (\text{Gen}, \text{Sig}, \text{Ver})$ consists of randomized key generation $(\mathsf{sk}, \mathsf{vk}) \leftarrow \text{Gen}(1^\lambda)$, signing $\sigma \leftarrow \text{Sig}(\mathsf{sk}, m)$, and verification, $0/1 \leftarrow \text{Ver}(\mathsf{vk}, \sigma, m)$. The (error-free) correctness condition asserts that for all λ and all messages m, we have

$$\Pr\left[(\mathsf{sk}, \mathsf{vk}) \leftarrow \text{Gen}(1^\lambda),\ \sigma \leftarrow \text{Sig}(\mathsf{sk}, m) : \text{Ver}(\mathsf{vk}, \sigma, m) = 1\right] = 1.$$

Definition 1 (Secure one-time Signatures). *A scheme $\mathcal{S} = (\text{Gen}, \text{Sig}, \text{Ver})$ is strongly existentially unforgeable under one-time attack if any PPT adversary $A = (A_1, A_2)$ has at most a negligible probability $\mathsf{negl}(\lambda)$ of winning the following game:*

1. $(\mathsf{sk}, \mathsf{vk}) \leftarrow \text{Gen}(1^k)$;
2. $(m, \mathsf{state}) \leftarrow A_1(\mathsf{vk})$;
3. $\sigma \leftarrow \text{Sig}(\mathsf{sk}, m)$;
4. $(m', \sigma') \leftarrow A_2(\mathsf{state}, \sigma)$.

A wins the game if $(m', \sigma') \neq (m, \sigma)$ but $\text{Ver}(\mathsf{vk}, \sigma', m') = 1$.

[5] Intuitively, multiple-ciphertext functional re-encryption oracle can be simulated by a single-ciphertext non-functional re-encryption oracle, by first homomorphically applying the function f "inside the encryption", and then calling the simpler oracle to ensure the resulting encryption is "fresh".

Encryption schemes. We recall below different notions of security for public-key encryption schemes. Such a scheme $\mathcal{E} = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ (over message space M which could depend on the security parameter) consists of:

- Key Generation algorithm $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathrm{Gen}(1^\lambda)$. Here dk is the secret key and ek is the public key.
- Encryption algorithm $\mathsf{ct} \leftarrow \mathrm{Enc}(\mathsf{ek}, \mathsf{pt})$ converting message $\mathsf{pt} \in M$ into ciphertext ct; and
- Decryption algorithm $\mathsf{pt} \leftarrow \mathrm{Dec}(\mathsf{dk}, \mathsf{ct})$ recovering the plaintext $\mathsf{pt} \in M \cup \{\perp\}$ from the ciphertext ct, where $\perp$ denotes a decryption failure.

The (error-free[6]) correction condition asserts that for all λ and all $\mathsf{pt} \in M$, we have

$$\Pr\left[(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathrm{Gen}(1^\lambda),\ \mathsf{ct} \leftarrow \mathrm{Enc}(\mathsf{ek}, \mathsf{pt}) : \mathrm{Dec}(\mathsf{dk}, \mathsf{ct}) = \mathsf{pt}\right] = 1.$$

Definition 2 ((nm)CPA/CCA1/CCA2/(multi)FuncCPA Security)**.** *An encryption scheme* $\mathcal{E} = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ *is* X*-secure for security notion*

$$X \in \{\mathit{CPA},\ \ \mathit{CCA1},\ \ \mathit{CCA2},\ \ \mathit{FuncCPA},\ \ 1\text{-}\mathit{FuncCPA},\ \mathit{ReEncCPA},\ \mathit{nmCPA}\},$$

if any PPT adversary $A = (A_1, A_2)$ *with access to oracles* $(\mathcal{O}_1, \mathcal{O}_2)$ *below has at most a negligible advantage* $\mathsf{negl}(\lambda)$ *in the following game:*

1. $b \leftarrow \{0,1\}$; $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathrm{Gen}(1^\lambda)$;
2. $(\mathsf{pt}_0, \mathsf{pt}_1, \mathsf{state}) \leftarrow A_1^{\mathcal{O}_1}(\mathsf{ek})$;
3. $\mathsf{ct} \leftarrow \mathrm{Enc}(\mathsf{ek}, \mathsf{pt}_b)$;
4. $b' \leftarrow A_2^{\mathcal{O}_2}(\mathsf{state}, \mathsf{ct})$.

The advantage is defined as $|\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]|$, *where the oracles* $(\mathcal{O}_1, \mathcal{O}_2)$ *are instantiated as follows for each notion* X*:*

CPA: $\mathcal{O}_1, \mathcal{O}_2$ always return $\perp$.

CCA1: $\mathcal{O}_1(\mathsf{ct}') = \mathrm{Dec}(\mathsf{dk}, \mathsf{ct}')$ is decryption oracle, and $\mathcal{O}_2$ always returns $\perp$.

CCA2: $\mathcal{O}_1(\mathsf{ct}') = \mathrm{Dec}(\mathsf{dk}, \mathsf{ct}')$ is decryption oracle, and $\mathcal{O}_2$ is the same as $\mathcal{O}_1$, except it returns $\perp$ on the challenge ct from Step 3 above.

FuncCPA: $\mathcal{O}_1(\mathsf{ct}'_1, \ldots, \mathsf{ct}'_\ell, f) = \mathcal{O}_2(\mathsf{ct}'_1, \ldots, \mathsf{ct}'_\ell, f) = \mathrm{Enc}(\mathsf{ek}, f(\mathrm{Dec}(\mathsf{dk}, \mathsf{ct}'_1), \ldots, \mathrm{Dec}(\mathsf{dk}, \mathsf{ct}'_\ell)))$ are multi-input functional re-encryption oracles, where $\ell \in \mathbb{Z}$ can be arbitrary and $f : (M \cup \{\perp\})^\ell \to M$ is any function (specified as a circuit).

1-FuncCPA: Same as FuncCPA, but all functions f are single input ($\ell = 1$);
$\mathcal{O}_1(\mathsf{ct}', f) = \mathcal{O}_2(\mathsf{ct}', f) = \mathrm{Enc}(\mathsf{ek}, f(\mathrm{Dec}(\mathsf{dk}, \mathsf{ct}')))$.

ReEncCPA: Same as FuncCPA, but all functions f are the identity $f(\mathsf{pt}) = \mathsf{pt}$;
$\mathcal{O}_1(\mathsf{ct}') = \mathcal{O}_2(\mathsf{ct}') = \mathrm{Enc}(\mathsf{ek}, \mathrm{Dec}(\mathsf{dk}, \mathsf{ct}'))$

nmCPA: $\mathcal{O}_1$ always returns $\perp$, while $\mathcal{O}_2$ accepts a single "parallel" query $\{(\mathsf{ct}'_i)\}$, and returns $\{\mathsf{pt}'_i\}$, after which it returns $\perp$ for all subsequent queries. As with CCA2 notion, $\mathsf{pt}'_i = \perp$, if $\mathsf{ct}'_i = \mathsf{ct}$ from Step 3; and otherwise it is the regular decryption oracle $\mathsf{pt}'_i = \mathrm{Dec}(\mathsf{dk}, \mathsf{ct}'_i)$.

[6] All the results in this work apply out-of-the-box also to schemes with decryption errors, as long as they only occur with negligible probability. Otherwise one can amplify correctness of the underlying CPA-secure scheme before applying our transformation.

Multiple-Keys Tag-Based Non-Malleability. For our transformation, it will be slightly more convenient to use a slight extension of nmCPA security notion, which is easily seen equivalent to the traditional nmCPA security given in Definition 2.

First, we will use a *tagged* nmCPA encryption, where the encryption and decryption routines are also given a tag tg, and correctness in only ensured when the same tag is used in both.

$$\Pr\left[(\mathsf{dk},\mathsf{ek}) \leftarrow \mathrm{Gen}(1^\lambda),\ \mathsf{ct} \leftarrow \mathrm{Enc}(\mathsf{ek},\mathsf{tg},\mathsf{pt}) : \mathrm{Dec}(\mathsf{dk},\mathsf{tg},\mathsf{ct}) = \mathsf{pt}\right] = 1.$$

Furthermore, the non-malleability security game is modified so that the adversary submits a set of tag/ciphertext pairs $\{(\mathsf{tg}'_i,\mathsf{ct}'_i)\}$ in parallel, such that each pair $(\mathsf{tg}'_i,\mathsf{ct}'_i) \neq (\mathsf{tg},\mathsf{ct})$ differs from the challenge tag/ciphertext pai from Step 3, and the oracle responds with $\mathsf{pt}'_i = \mathrm{Dec}(\mathsf{dk},\mathsf{tg}'_i,\mathsf{ct}'_i)$. Notice, a tagged scheme can always be converted into a non-tagged scheme by just omitting the tag. Conversely, non-tagged scheme for large message space can be made to support tags, by viewing the tag as part of the message, and then checking that the decrypted tag matches the declared one.

Second, we will use a multiple-keys/multiple-message nmCPA which will be slightly more convenient for our proof. This is known to be equivalent to the notion from definition 2, e.g., [PSV06, Thm 4]. The full definition is given below.

Definition 3 (tag-nmCPA Security)**.** *A tagged scheme* $\mathcal{E} = (\mathrm{Gen},\mathrm{Enc},\mathrm{Dec})$ *is tag-non-malleable secure if for any polynomial* $p(\cdot)$*, a* conforming *PPT adversary* $A = (A_1, A_2, A_3)$ *has at most a negligible advantage* $\mathsf{negl}(\lambda)$ *in the following game:*

1. $b \leftarrow \{0,1\}$; $(\mathsf{dk}_i,\mathsf{ek}_i) \leftarrow \mathrm{Gen}(1^\lambda)$ *for* $i = 1,2,\ldots,p(\lambda)$;
2. $\left((i_1,\mathsf{pt}_1^0,\mathsf{pt}_1^1,\mathsf{tg}_1),\ldots,(i_m,\mathsf{pt}_m^0,\mathsf{pt}_m^1,\mathsf{tg}_m),\mathsf{state}\right) \leftarrow A_1(\{\mathsf{ek}_i\})$;
3. $\mathsf{ct}_j \leftarrow \mathrm{Enc}(\mathsf{ek}_{i_j},\mathsf{tg}_j,\mathsf{pt}_j^b)$ *for* $j = 1,\ldots,m$;
4. $\left((k_1,\mathsf{ct}'_1,\mathsf{tg}'_1),\ldots,(k_n,\mathsf{ct}'_n,\mathsf{tg}'_n),\mathsf{state}'\right); \leftarrow A_2(\mathsf{state},\mathsf{ct}_1,\ldots,\mathsf{ct}_m)$;
5. $\mathsf{pt}'_\ell \leftarrow \mathrm{Dec}(\mathsf{dk}_{k_\ell},\mathsf{tg}'_\ell,\mathsf{ct}'_\ell)$ *for* $\ell = 1,\ldots,n$;
6. $b' \leftarrow A_3(\mathsf{state}',\mathsf{pt}'_1,\ldots,\mathsf{pt}'_n)$.

A is conforming if the pairs $\{(k_\ell,\mathsf{tg}'_\ell,\mathsf{ct}'_\ell) : \ell = 1,\ldots,n\}$ *are disjoint from* $\{(i_j,\mathsf{tg}_j,\mathsf{ct}_j) : j = 1,\ldots,m\}$*. The advantage is defined as*

$$|\Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]|.$$

2.1 An Alternative Definition of FuncCPA

When proving the FuncCPA security of our construction, it is convenient to use a possibly-stronger notion than definition 2, that we call FuncCPA$^+$. Rather than requiring that the functional re-encryption oracle does not help in breaking semantic security, this definition states that functional re-encryption oracle does not help because it cannot by distinguished from fresh encryptions of 0 (which the attacker can do itself).[7]

Definition 4 (FuncCPA$^+$ Security)**.** *A (non-tagged) scheme* $\mathcal{E} = (\mathrm{Gen},\mathrm{Enc},\mathrm{Dec})$ *is FuncCPA*$^+$*-secure if any PPT adversary A with access to a re-encryption oracle has at most a negligible advantage* $\mathsf{negl}(\lambda)$ *in the following game:*

$$1.\ b \leftarrow \{0,1\};\ (\mathsf{dk},\mathsf{ek}) \leftarrow \mathrm{Gen}(1^\lambda); \qquad 2.\ b' \leftarrow A^{\mathsf{reEnc}_b(\mathsf{dk},\mathsf{ek},\cdot,\cdot)}(\mathsf{ek}).$$

where the re-encryption oracle takes an arbitrary ℓ *and* $f : (M \cup \{\perp\})^\ell \to M$ *(given as a circuit):*

$$\mathsf{reEnc}_b(\mathsf{dk},\mathsf{ek},\mathsf{ct}_1,\ldots,\mathsf{ct}_\ell,f) = \begin{cases} \mathsf{E}(\mathsf{ek}, f(\mathrm{Dec}(\mathsf{dk},\mathsf{ct}_1),\ldots,\mathrm{Dec}(\mathsf{dk},\mathsf{ct}_\ell))) & \textit{if } b = 1 \\ \mathsf{E}(\mathsf{ek}, 0) & \textit{if } b = 0 \end{cases}$$

[7]Having two such flavors is reminiscent of definitions of circular security: Over there one notion asserts that an encryption of the secret key does not help the attacker violate semantic security, and the other requires that the attacker cannot distinguish such encryption from an encryption of zero.

The advantage is defined as $|\Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]|$. *We can also define restricted notions* 1-*FuncCPA*$^+$ *and ReEncCPA*$^+$*, corresponding to single input* f $(\ell = 1)$ *and the identity function* f*, respectively.*

First, we show that this notion indeed implies FuncCPA security.

Lemma 1. *Any scheme which is FuncCPA*$^+$*-secure, is also FuncCPA-secure. (Analogously, the same holds for restricted notions* 1-*FuncCPA and ReEncCPA.)*

Proof. We first recall that the "left-or-right" notion from definition 2 where the attacker chooses $\mathsf{pt}_0, \mathsf{pt}_1$ and gets an encryption of one of them, is known to be equivalent to a "real-or-zero" notion of security where the attacker only chooses pt_1, and gets either an encryption of pt_1 or an encryption of zero. (These are equivalent upto a factor of 2 in the advantage.) It is therefore sufficient to show that definition 4 implies this real-or-zero notion.

Let $\mathcal{E}$ be a scheme satisfying definition 4, and we want to show that it also satisfy the (real-or-zero variant of) definition 2. Let A be an adversary with access to a functional re-encryption oracle, and we want to show that that it only has a negligible advantage in the real-or-zero game against $\mathcal{E}$. Consider the probability of A outputting 1 in the following four experiments:

1. A's oracle is implemented by a true functional re-encryption oracle, and its challenge-ciphertext query is answered by an encryption of pt_1.
2. A's oracle is implemented by a zero-encrypting oracle, and its challenge-ciphertext query is answered by an encryption of pt_1.
3. A's oracle is implemented by a zero-encrypting oracle, and its challenge-ciphertext query is answered by an encryption of 0.
4. A's oracle is implemented by a true functional re-encryption oracle, and its challenge-ciphertext query is answered by an encryption of 0.

The probabilities in Experiments 1 vs. 2 are close (upto a negligible difference) by the FuncCPA$^+$-security of $\mathcal{E}$, and the same holds for the probabilities in Experiments 3 vs. 4. Moreover, the probabilities in Experiments 2 vs. 3 are close (upto a negligible difference) by the CPA-security of $\mathcal{E}$, which is implied by FuncCPA$^+$-security. Hence the probability of A outputting 1 in experiments 1 vs. 4 are close upto a negligible difference, as needed.

The proof for restricted variants follows identically. □

Are These Definitions Equivalent? Lemma 1 says that FuncCPA$^+$ implies FuncCPA, but we do not know if there is also an implication in the other direction. One piece of evidence that points toward FuncCPA$^+$ being strictly stronger than FuncCPA, is that we can show a separation for the analogous notions with a *non-functional* re-encryption oracle; namely, ReEncCPA notion. In this notion, a re-encryption query consists of only a ciphertexts ct (without the function f) and it is answered by $\mathsf{ct}' = \mathrm{Enc}(\mathrm{Dec}(\mathsf{ct}))$. See Lemma 10.

Other (non-)Relations. In Appendix A we also have several other observations of interest. In Lemma 8, we also show that a CCA1-secure scheme is always FuncCPA$^+$-secure. In Lemma 11 we also show that a (single-input) 1-FuncCPA$^+$-secure scheme is not always (multi-input) FuncCPA-secure, while in Lemma 9 we show that nmCPA-security of a given scheme is incomparable with any of the FuncCPA/1-FuncCPA/ReEncCPA-securities (enhanced or not) of that scheme.

3 From CPA to FuncCPA

This section is devoted to proving our main Theorem 1. As mentioned, we will actually prove (potentially) stronger FuncCPA$^+$ security; Namely:

Theorem 2. *If CPA-secure encryption schemes exist, then so do FuncCPA$^+$-secure encryption schemes. Moreover, the transformation can be made black-box in the underlying CPA-secure scheme.*

As described in the introduction, we show that applying the CPA-to-nmCPA black-box transformation of Choi *et al.* [CDMW18a], to a scheme *which is already nmCPA-secure*, results in a FuncCPA$^+$ scheme. As the existence of CPA-secure schemes implies the existence of nmCPA-secure ones [PSV06], even with a black-box transformation [CDMW18a], the theorem follows.

For simplicity of notation (e.g., to avoid double indices), we first describe our proof for single-input re-encryption queries (i.e., 1-FuncCPA notion, corresponding to single-input function f with arity $\ell = 1$). However, there is basically no difference in extending the proof to support multiple ciphertext queries (e.g., general $\ell \geq 1$), and we sketch the extension from 1-FuncCPA$^+$ to full FuncCPA$^+$ security of our transformation in Section 3.5.

3.1 Technical Overview

Before describing the CDMW construction itself, we highlight the abstract properties that it satisfies, and how they are used to prove FuncCPA security. We rely on the following properties:

1. We have a notion of valid/invalid ciphertexts, and all ciphertexts output by encryption are valid.
2. For any challenge ciphertext ct^*, the reduction is able to find an alternate "somewhat defective" secret key, which decrypts *all valid ciphertexts except the challenge ciphertext* ct^* identically to the original key, but is incapable of breaking the semantic security of ct^*.
3. An adversary who only sees the public key, cannot produce an invalid ciphertext that decrypts to anything but 0, via either the original key or any defective key.

Non-Malleability. Let us briefly outline how the above properties are used in CDMW to show non-malleability: First, they switch to using an alternate "somewhat defective" decryption key from (2) to answer decryption queries. They argue that all decrpytion queries are answered identically with this change. As per (2), this is true for decryption queries with a valid ciphertext, and as per (3), queries with invalid ciphertext are always answered by 0 in both cases. We note that this step crucially relies on the adversary making only one (parallel) decryption query rather than many adaptive queries; i.e. it only provides non-malleability rather than CCA security. Indeed, an adversary making multiple adaptive decryption queries (with valid ciphertexts) can learn information about the secret key from answers to previous decryption queries, so can no longer rely on (3).[8] Second, after switching to using a "somewhat defective" decryption key, they can switch the challenge ciphertext ct^* from an encryption of pt_0 to an encryption of pt_1 by relying on property (2) that semantic security of ct^* is preserved even given the alternate decryption key.

FuncCPA$^+$ Security. Now we show how to use the above properties to prove FuncCPA$^+$ security. We define a *bad* event that the adversary submits an invalid ciphertext that does not decrypt to 0 (either by the original or any alternate decryption key) during the course of the game. As long as the bad event does not happen, we can replace the output of each functional re-encryption query (one by one) by an encryption of 0 via the same argument as above. On the other hand, we argue that the probability of the bad event occurring is negligible: To cause the bad event to happen for the first time on functional re-encryption query i, the adversary would have needed to learn something about the secret key from the first $i-1$ functional re-encryption queries. But because the bad event did not occur during those queries yet, we can replace

[8]Valid ciphertexts are not necessarily correctly generated and may decrypt differently depending on the secret key.

their outputs by encryptions of 0 (by the same argument as above) and argue that this cannot change the probability of the bad event occurring for the first time in the i'th query. Once the first $i-1$ queries return 0 and do not induce the bad event, we know that they do not reveal anything about the secret key, which ensures that probability of the bad event happening on the ith query is negligible.

Formalizing this argument, however, requires handling the following subtle point: when changing a ciphertext to an encryption of 0 in each step, we must show this change is not only imperceptible to the adversary, but it also does not affect the probability of the bad event. In particular, the reduction must check if the bad event occurred at the very end of the game, which requires checking if various ciphertexts submitted by the adversary during the game were valid or invalid. Therefore, we need a stronger version of property (2) to hold: the semantic security of ct^* holds even given the alternate decryption key *and a single parallel query to a valid/invalid ciphertext check.* To achieve this stronger property, we need the underlying component scheme used by this transformation to already be non-malleable.

3.2 Building Blocks

Non-malleability. Recall from Definition 2 that (tagged) non-malleable encryption is a weaker variant of (tagged) CCA-secure encryption, where the adversary only gets to make a single non-adaptive query to the decryption oracle (but that query can ask to decrypt many ciphertexts). Below it will be convenient to use the multi-key/multi-message variant of this notion (cf. Definition 3). As mentioned earlier, this is known to be equivalent to the single-key variant from Definition 2 (e.g., [PSV06, Thm 1]).[9]

One-time Signatures. Our main construction also uses (one-time) signatures, which are strongly existentially unforgeable, as per Definition 1.

Secret Sharing Encoding Schemes. We will also use the notion of secret-sharing encoding scheme $\mathcal{C} = (\mathsf{E}, \mathsf{D})$, similar to the notion of linear error-correcting secret-sharing [CDM+20]. Such a scheme comes with efficient randomized encoding E and decoding D, and is parameterized by underlying symbol space Σ, as well as integers k (dimension), n (length), d (decoding radius), and t (privacy parameter). We sometimes abuse notations, denoting by $\mathcal{C}$ the resulting code itself (i.e., the image of the encoding routine).

Let $\left[\binom{n}{t}\right]$ denote the collection of all the subsets $S \subset [n]$ of cardinality t, the code $\mathcal{C}$ has the following features:

- The encoding is $\mathsf{E} : \Sigma^k \times \mathcal{R} \to \Sigma^n$, where $\mathcal{R}$ is the randomness space.

  The decoding is $\mathsf{D} : \Sigma^n \to ((\Sigma^k \times \mathcal{R}) \cup \{\perp\})$, such that $\forall x \in \Sigma^k, r \in \mathcal{R}$, we have $\mathsf{D}(\mathsf{E}(x,r)) = (x,r)$.

  Below when we say "decoding to a codeword", we mean a procedure $\mathsf{D}' : \Sigma^n \to (\Sigma \cup \{\perp\})^n$ which is defined as

$$\mathsf{D}'(z \in \Sigma^n) = \begin{cases} \perp^n & \text{if } \mathsf{D}(z) = \perp \\ \mathsf{E}(\mathsf{D}(z)) & \text{otherwise.} \end{cases}$$

- The decoding radius of $\mathcal{C}$ is at least some large enough d (see below). Namely, for any $x \in \Sigma^k, r \in \mathcal{R}$ and any word $z \in \Sigma^n$ of Hamming distance at most d from $\mathsf{E}(x,r)$, it holds that $\mathsf{D}(z) = (x,r)$.

- There is an efficient extension procedure $\text{Extend} : \Sigma^k \times \Sigma^t \times \left[\binom{n}{t}\right] \to \Sigma^n$ that take as input $x \in \Sigma^k, y \in \Sigma^t$, and a size-$t$ subset $S \subset [n]$, $|S| = t$, and outputs a codeword $z \in \mathcal{C}$ such that $\mathsf{D}(z) = (x,r)$ for some r, and $z|_S = y$ (i.e., the symbols of z in positions from S are exactly y).

  Moreover, for any $x \in \Sigma^k$ and any $S \in \left[\binom{n}{t}\right]$, the following two distributions are equal:

$$\{r \leftarrow \mathcal{R} : \text{ output } \mathsf{E}(x,r)\} \text{ and } \{y \leftarrow \Sigma^t : \text{ output Extend}(x,y,S)\}.$$

[9]The theorem in [PSV06, Thm 4] is stated for a non-tagged scheme, but it holds equally for the tagged version.

The parameters of $\mathcal{C}$ are set to ensure that $(1-\frac{d}{n})^t \le 2^{-\lambda}$. [10]

Some examples of such codes: using Shamir secret-sharing we can get a construction over a large enough field $\Sigma = \mathbb{F}_{2^\ell}$ with (say) $k=1$, $t=2\lambda$, $n=3t$ and $d=t$. Or we can encode the Shamir-based constructions in binary, using $\Sigma=\{0,1\}$, $k \ge 3+\log\lambda$, $t=2\lambda k$, $n=3t$ and $d=t$. A more general use of Reed-Solomon codes (still with large enough $\Sigma=\mathbb{F}_{2^\ell}$) could be an arbitrary k, $t=2\lambda$ and $n=3(t+k-1)$, and $d=t+k-1$. One can also get better efficiency using Algebraic-Geometric codes as described in [CDM+20].

3.3 The CDMW Transformation

We start by describing the CDMW transformation, using an abstraction similar to Coretti *et al.* [CDM+20]. Denote the security parameter by λ, and we want to construct an encryption scheme with message space Σ^k. Below we assume that $0 \in \Sigma$, and we sometimes think of Σ as a large enough field. The construction uses the following components

- An underlying encryption scheme $\mathcal{E} = (\mathrm{Gen}_E, \mathrm{Enc}, \mathrm{Dec})$ with message space Σ, satisfying tag-nmCPA security (Definition 3). Below we sometimes call it the *component encryption scheme.*
- A one-time signature scheme $\mathcal{S} = (\mathrm{Gen}_S, \mathrm{Sig}, \mathrm{Ver})$, satisfying strong existential unforgeability (Definition 1). We denote by $\kappa = \kappa(\lambda)$ the size of the verification key.
- A secret-sharing encoding scheme $\mathcal{C} = (\mathsf{E}, \mathsf{D})$, with underlying symbol space Σ, dimension k, length n, decoding radius d, and privacy parameter t.

The CDMW construction is an encryption scheme for messages $\mathsf{pt} \in \Sigma^k$, $\mathcal{E}' = (\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}')$ as follows:

Key generation $\mathrm{Gen}'(1^\lambda)$**.**

1. Generate $2\kappa n$ key pairs $(\mathsf{ek}_{i,j,b}, \mathsf{dk}_{i,j,b}) \leftarrow \mathrm{Gen}_E(1^\lambda)$ with $i \in [\kappa], j \in [n]$, and $b \in \{0,1\}$;
2. Choose at random a size-t subset $S^* \in \left[\binom{n}{t}\right]$, and a random row $i^* \leftarrow [\kappa]$;

The public key consists of all the $2\kappa n$ component public keys, and the secret key consists of the $2(n+(\kappa-1)t)$ component secret keys for the designated row i^* and columns $j \in S^*$,

$$\mathsf{ek}' = \left\{\mathsf{ek}_{i,j,b} : b \in \{0,1\}, i \in [\kappa], j \in [n], \right\}$$

$$\mathsf{dk}' = \left(i^*, S^*, \left\{\mathsf{dk}_{i,j,b} : b \in \{0,1\}, i = i^* \text{ or } j \in S^*\right\}\right).$$

Encryption $\mathrm{Enc}'(\mathsf{ek}', \mathsf{pt})$**,** $\mathsf{pt} \in \Sigma^k$**.**

1. Choose a signature key pair $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathrm{Gen}_S(1^\lambda)$, with $|\mathsf{vk}| = \kappa$. Denote the i'th bit in vk by v_i.
2. Choose encoding randomness $r \leftarrow \mathcal{R}$ and compute the codeword $c := \mathsf{E}(\mathsf{pt}, r) \in \mathcal{C}$.
3. For all $i \in [\kappa], j \in [n]$ encrypt $c_j \in \Sigma$ under ek_{i,j,v_i} with tag vk: $\mathsf{ct}_{i,j} \leftarrow \mathrm{Enc}(\mathsf{ek}_{i,j,v_i}, \mathsf{vk}, c_j)$.
   Denote the concatenation of all these κn component ciphertexts by $\vec{\mathsf{ct}} = (\mathsf{ct}_{i,j} : i \in [\kappa], j \in [n])$.
4. Compute the signature $\sigma \leftarrow \mathrm{Sig}(\mathsf{sk}, \vec{\mathsf{ct}})$.

The compound ciphertext is $\mathsf{ct}' = \left(\vec{\mathsf{ct}}, \mathsf{vk}, \sigma\right)$.

[10] We note that the requirements from Extend imply that t cannot be too close to n, at the very least we need $n \ge t+k$ so that any t-symbol string can be extended to an encoding of any k-symbol information word.

Decryption $\mathrm{Dec}'(\mathsf{dk}', \mathsf{ct}')$. Parse $\mathsf{ct}' = \big(\vec{\mathsf{ct}}, \mathsf{vk}, \sigma\big)$.

1. Check the signature, if $\mathrm{Ver}(\mathsf{vk}, \vec{\mathsf{ct}}, \sigma) = 0$ then output 0^k and halt.
2. Decrypt all the component ciphertexts for which you have keys, $\gamma'_{i,j} \leftarrow \mathrm{Dec}(\mathsf{dk}_{i,j,v_i}, \mathsf{vk}, \mathsf{ct}_{i,j})$ for $i = i^*$ or $j \in S^*$.
3. Let $c' = (\gamma'_{i^*,j} : j \in [n])$ be the word encoded in row i^*, and correct c' to a codeword, setting $\bar{c} = \mathsf{D}'(c')$.
4. Check that all the columns in S^* agree with $\bar{c}$: For all $i \in [\kappa], j \in S^*$, $\gamma'_{i,j} = \bar{c}_j$. If any of these checks fails then output 0^k and halt.
5. Decode $(x, r) := \mathsf{D}(\bar{c})$ and output x.

Connection to Technical Overview. Before giving a formal proof of security, we briefly discuss how this construction satisfies the abstract properties (1)-(3) from the technical overview in Section 3.1.

For (1), we define valid ciphertexts as ones where there is a single codeword $\bar{c} \in \mathcal{C}$ such that the component ciphertexts in each row i decrypt to a value sufficiently close (within distance d) to $\bar{c}$. Otherwise ciphertexts are invalid.

For (2), we can consider different decryption secret keys depending on the row i^* they decrypt. The original key picks one fixed row i^* in which it knows all the component secret keys for both bits $b \in \{0, 1\}$. The alternate "somewhat defective" decryption keys will only know all the secret keys for either $b = 0$ or $b = 1$ (but not both) in each row i. In particular, for a challenge cipherext ct^* with verification vk^* we will pick a somewhat defective decryption key such that, for each row i, it only knows the secret keys with bit $b = 1 - v_i^*$ where v_i^* is the i'th bit of vk^*. It will decrypt each ciphertext with verification key $\mathsf{vk} \neq \mathsf{vk}^*$ using the first row i in which the verification key bits differ $v_i \neq v_i^*$. In both cases, we also keep all the component secret keys in the special columns S^* and perform the same checks as the original decryption procedure. This ensures that: (a) the somewhat defective decryption key is incapable of breaking the semantic security of ct^* since it is only capable of decrypting the component ciphertexts of ct^* is the columns S^*, but these don't reveal anything about the message by the hiding of the secret sharing encoding, (b) the somewhat defective decryption key decrypts every valid ciphertext $\mathsf{ct} \neq \mathsf{ct}^*$ (having $\mathsf{vk} \neq \mathsf{vk}^*$) identically to the original key, since the row i it decrypts will decode to the same codeword $\bar{c}$ as in the original decryption and the checks performed are identical.

For (3), in order for the adversary to produce an invalid ciphertext that decrypts to anything but 0, (via either the original key or any defective key), there must be some row i that decrypts to a value c that decodes to some codeword $\bar{c}$, and some row i' (possibly $i' = i$) that decrypts to a value c' such that c' is too far (more than d distance) from $\bar{c}$. But in that case, the decryption procedure will output 0 with overwhelming probability over the choice of S^*; it only fails to do so if none of the columns of S^* overlap with any of the positions in which c' differs from $\bar{c}$, but this only happens with probability $\leq (1 - \frac{d}{n})^t \leq 2^{-\lambda}$.

3.4 Proof of Security

Recall, we first give our main proof for 1-FuncCPA$^+$ security (i.e., a single-input functional re-encryption oracle). However, as we then discuss in Section 3.5, the proof extends directly to general FuncCPA$^+$ security (with a multi-input functional re-encryption oracle), with only minimal changes.

Lemma 2. *If the component encryption scheme $\mathcal{E}$ satisfies tag-nmCPA security (Definition 3) and the signature $\mathcal{S}$ satisfies strong existential unforgeability (Definition 1), then the compound scheme $\mathcal{E}'$ above is 1-FuncCPA$^+$ secure.*

Namely we show that under the stated assumptions, the view of a FuncCPA$^+$ attacker A in the "real world" is indistinguishable from its view in an "ideal world", in which all the functional re-encryption queries are answered by encrypting the all-zero plaintext word 0^k.

Simplifying assumptions. Consider some 1-FuncCPA$^+$ adversary A. We sometimes refer to ciphertexts that A submits to the functional re-encryption oracle as *input ciphertexts*, and the ones returned from the oracle are called *output ciphertexts.*

Firstly, without loss of generality, we can assume that A never submits an input ciphertext which is equal to a prior output ciphertext. Indeed, if a previous functional re-encryption query (ct, f) returned some ct', then a new query (ct', f') could just as well be replaced by $(\mathsf{ct}, f' \circ f)$: By definition these two queries have identical answers.

Secondly, we will assume that A never queries the oracle on an input ciphertext $\mathsf{ct}' = \big(\vec{\mathsf{ct}}, \mathsf{vk}, \sigma\big)$ in which the signature fails to verify $\mathrm{Ver}(\mathsf{vk}, \vec{\mathsf{ct}}, \sigma) = 0$. This is because we can test this property efficiently given only ct' and can replace any such query (ct', f) with (ct'', f) where ct'' is a fresh (correctly generated) encryption of 0^k.

Thirdly, given the above assumptions, we can also assume (by reduction to the signature security) that the sets of verification keys in the input ciphertexts is disjoint from that in the output ciphertexts: To use the same verification key vk as in previous output ciphertext, the adversary will need to forge a signatures on the new $\vec{\mathsf{ct}}$ relative to that previous vk (or a new signature σ on the same $\vec{\mathsf{ct}}$), which can only happen with negligible probability.

We call an adversary for which the above three assumptions hold a *conforming adversary.* The arguments above imply that conforming adversaries have as much of an advantage as general ones in distinguishing the real game from the "ideal" one (upto negligible difference due to forgery of the signatures). Below we therefore fix one conforming adversary A and analyze its advantage. Important concepts in our analysis are *valid/invalid* ciphertexts and bad events, which are defined next.

Valid ciphertext. A compound ciphertext $\mathsf{ct}' = \big(\vec{\mathsf{ct}}, \mathsf{vk}, \sigma\big)$ is valid — *relative to all the component public/secret key pairs*, $\{(\mathsf{ek}_{i,j,b}, \mathsf{dk}_{i,j,b}) : b \in \{0,1\}, i \in [\kappa], j \in [n]\}$ — if the following condition holds:

- There exists a unique codeword $\bar{c} \in \mathcal{C}$ such that for all $i \in [\kappa]$, using the i'th row for decryption yields a word which is at most d away from $\bar{c}$. Namely, setting $\gamma_{i,j} := \mathrm{Dec}(\mathsf{dk}_{i,j,v_i}, \mathsf{vk}, \mathsf{ct}_{i,j})$ and denoting $c_i = (\gamma_{i,1}, \ldots, \gamma_{i,n})$ for all i, all the c_i's are of Hamming distance at most d from $\bar{c}$.

A ciphertext is *invalid* if it is not valid.

Bad events. A big part of the analysis below is devoted to bounding the probability of the bad event in which A submits an invalid ciphertext to the functional re-encryption oracle, but this invalid ciphertext "is not caught".

Consider the set S^* and all the component public/secret key pairs $\{(\mathsf{ek}_{i,j,b}, \mathsf{dk}_{i,j,b}) : b \in \{0,1\}, i \in [\kappa], j \in [n]\}$. We denote by Bad the event in which A makes a functional re-encryption query with an *invalid ciphertext* $\mathsf{ct}' = \big(\vec{\mathsf{ct}}, \mathsf{vk}, \sigma\big)$, but the check in step (4) of the decryption procedure does not trigger. Namely, for all $i \in [\kappa]$, denote by $c_i = (\gamma_{i,1}, \ldots, \gamma_{i,n})$ the decryption of the i'th row (as in the definition of valid ciphertexts above). Then the event Bad occurs if there is a codeword $\bar{c} = (\bar{\gamma}_1, \ldots, \bar{\gamma}_n) \in \mathcal{C}$ such that:

- There are indices $i_1, i_2 \in [\kappa]$, such that $\mathsf{D}'(c_{i_1}) = \bar{c}$, but c_{i_2} is at Hamming distance more than d from $\bar{c}$.
- All the c_i's agree with $\bar{c}$ on all the symbols in the columns $j \in S^*$: $\forall i \in [\kappa], \forall j \in S^* \ : \ \gamma_{i,j} = \bar{\gamma}_j$.

Let q be a polynomial upper bound on the number of functional re-encryption queries made by A. For all $u \in [q]$ denote by Bad_u the event in which the u'th functional re-encryption query causes the above bad event to occur. Also, for any $v \in [q]$, denote by $\mathsf{1stBad}_v$ the event where the first query to cause the bad event to occur is the v'th query i.e.,:

$$\mathsf{1stBad}_v = \overline{\mathsf{Bad}_1} \ \& \ \ldots \ \& \ \overline{\mathsf{Bad}_{v-1}} \ \& \ \mathsf{Bad}_v.$$

Hybrids. With the same bound q on the number of decryption queries, we consider a set of $q+1$ hybrid experiments, $H_0, H_1, \ldots, H_q$. In the hybrid H_u the first u functional re-encryption queries are answered by encrypting the all-zero plaintext, and all the queries from $u+1$ and on are answered as in the real 1-FuncCPA$^+$ game. (Note that the notions of valid/invalid input ciphertexts and bad events apply to all these hybrids.) The real 1-FuncCPA$^+$ game is therefore H_0, the ideal game is H_q, and proving Lemma 2 boils down to showing that

$$\left|\Pr_{H_q}[A \to 1] - \Pr_{H_0}[A \to 1]\right| \le \mathsf{negl}(\lambda), \tag{1}$$

for some negligible function $\mathsf{negl}(\cdot)$, where $A \to 1$ is the event of A halting after outputting 1. To establish Equation (1), we first note that

$$\begin{aligned}\left|\Pr_{H_q}[A \to 1] - \Pr_{H_0}[A \to 1]\right| \quad \le \quad & \left| \Pr_{H_q}[A \to 1 \ \& \ \overline{\mathsf{Bad}}] - \Pr_{H_0}[A \to 1 \ \& \ \overline{\mathsf{Bad}}] \right| \\ & + \Pr_{H_0}[\mathsf{Bad}] + \Pr_{H_q}[\mathsf{Bad}].\end{aligned}$$

The heart of proof below consists of the following three lemmas, whose proofs are provided later in this section:

Lemma 3. *For all $u, v \in [q]$ there is a negligible function $\mathsf{negl}(\cdot)$ such that*

$$\left|\Pr_{H_{u-1}}[\mathsf{1stBad}_v] - \Pr_{H_u}[\mathsf{1stBad}_v]\right| < \mathsf{negl}(\lambda).$$

Lemma 4. *For all $v \in [q]$ there is a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr_{H_q}[\mathsf{Bad}_v] < \mathsf{negl}(\lambda)$.*

Lemma 5. *For all $u \in [q]$ there is a negligible function $\mathsf{negl}(\cdot)$ such that*

$$\left|\Pr_{H_u}[A \to 1 \ \& \ \overline{\mathsf{Bad}}] - \Pr_{H_{u-1}}[A \to 1 \ \& \ \overline{\mathsf{Bad}}]\right| \le \mathsf{negl}(\lambda).$$

Given these three lemmas, we complete the proof as follows:

$$\begin{aligned}\left|\Pr_{H_q}[A \to 1] - \Pr_{H_0}[A \to 1]\right| \quad & \le \quad \left| \Pr_{H_q}[A \to 1 \ \& \ \overline{\mathsf{Bad}}] - \Pr_{H_0}[A \to 1 \ \& \ \overline{\mathsf{Bad}}] \right| \\ & \qquad + \Pr_{H_0}[\mathsf{Bad}] + \Pr_{H_q}[\mathsf{Bad}] \\ & \le \quad \sum_{u=1}^{q} \left| \Pr_{H_u}[A \to 1 \ \& \ \overline{\mathsf{Bad}}] - \Pr_{H_{u-1}}[A \to 1 \ \& \ \overline{\mathsf{Bad}}] \right| \\ & \qquad + \sum_{v=1}^{q} \Pr_{H_0}[\mathsf{1stBad}_v] + \sum_{v=1}^{q} \Pr_{H_q}[\mathsf{Bad}_v] \\ & \le \quad 2q \cdot \mathsf{negl}(\lambda) + \sum_{v=1}^{q} \Pr_{H_0}[\mathsf{1stBad}_v],\end{aligned} \tag{2}$$

where in the first inquelity we rely on $\mathsf{Bad} = \bigvee_{u\in[q]} \mathsf{Bad}_u = \bigvee_{u\in[q]} \mathsf{1stBad}_u$ and the last inequality is due to Lemmas 4 and 5. Moreover, for any $v \in [q]$ we have

$$\begin{aligned}\Pr_{H_0}[\mathsf{1stBad}_v] \quad & = \quad \Pr_{H_q}[\mathsf{1stBad}_v] + \sum_{u=1}^{q} \left(\Pr_{H_{u-1}}[\mathsf{1stBad}_v] - \Pr_{H_u}[\mathsf{1stBad}_v] \right) \\ & \le \quad \Pr_{H_q}[\mathsf{Bad}_v] + \sum_{u=1}^{q} \left| \Pr_{H_{u-1}}[\mathsf{1stBad}_v] - \Pr_{H_u}[\mathsf{1stBad}_v] \right| \\ & \le \quad (q+1) \cdot \mathsf{negl}(\lambda),\end{aligned} \tag{3}$$

with the last inequality due to Lemmas 3 and 4. Plugging the expression from Eq. (3) into Eq. (2), we get

$$\left|\Pr_{H_q}[A \to 1] - \Pr_{H_0}[A \to 1]\right| \leq 2q \cdot \mathsf{negl}(\lambda) + q \cdot ((q+1) \cdot \mathsf{negl}(\lambda)) = (q^2 + 3q) \cdot \mathsf{negl}(\lambda).$$

Since $\mathsf{negl}(\cdot)$ is negligible and q is polynomial, then also $(q^2 + 3q) \cdot \mathsf{negl}(\cdot)$ is negligible, completing the proof of Lemma 2. □

3.4.1 Proving Lemmas 3 through 5

The proofs make use of the following two easy observations:

Fact 6. *For any $u \geq v$, the v'th oracle query in hybrid H_u is answered in a way that does not depend of the index i^* or the set S^* in the secret key.*

Proof. By definition of H_u, the v'th output ciphertext consists of just encryption of the all-zero plaintext, regardless of anything else. □

Fact 7. *For any $u, v \in [q]$, if the event Bad_v does not occur in the hybrid H_u, then the v'th oracle query is answered in a way that does not pendent of the index i^* in the secret key.*

Proof. Follows by definition of the bad event Bad_v. If Bad_v does not occur, the v'th input ciphertext is either a valid ciphertext (does not satisfy the first condition of the bad event), or an invalid ciphertext that triggers one of the checks on decryption (does not satisfy the second condition of the bad event). In the first case, all the rows i are decrypted to a word c_i within the decoding radius d of the code from the same codeword $\bar{c}$, so the recovered value $\bar{c}$ in step (3) of decryption will be the same no matter which row i is used, and the rest of the decryption procedure does not depend on i. On the other hand, in the second case, the checks in step (4) of decryption will be triggered and cause the decrypted value to be 0^k no matter which row i is used. □

Truncated Hybrids. When analyzing the events Bad_v or $\mathsf{1stBad}_v$, it is convenient to consider *truncated hybrids*, where the game is aborted as soon as A makes the v'th query, indeed whether or not Bad_v or $\mathsf{1stBad}_v$ happen is fully determined as soon as A made that query, so there is no reason to continue the game. Below we denote by $H_u|_v$ the hybrid H_u, truncated immediately after A's v'th functional re-encryption query.

3.4.2 Proof of Lemma 3

Lemma 3. *For all $u, v \in [q]$ there is a negligible function $\mathsf{negl}(\cdot)$ such that*

$$\left|\Pr_{H_{u-1}}[\mathsf{1stBad}_v] - \Pr_{H_u}[\mathsf{1stBad}_v]\right| < \mathsf{negl}(\lambda).$$

Proof. By definition of the truncated hybrids, for any $u, v \in [q]$ we have that $\Pr_{H_u|_v}[\mathsf{1stBad}_v] = \Pr_{H_u}[\mathsf{1stBad}_v]$. Moreover, we note that when $u \geq v$, all the functional re-encryption queries in $H_u|_v$ are answered with encryption of the all-zero plaintext. It follows that when $u > v$ then the hybrids $H_{u-1}|v$ and $H_u|_v$ are identical, and therefore

$$\Pr_{H_{u-1}}[\mathsf{1stBad}_v] = \Pr_{H_{u-1}|_v}[\mathsf{1stBad}_v] = \Pr_{H_u|_v}[\mathsf{1stBad}_v] = \Pr_{H_u}[\mathsf{1stBad}_v].$$

It remains to prove Lemma 3 for $u \leq v$, which we do by reduction to the tag-nmCPA security of the component encryption scheme $\mathcal{E}$. Before giving a detailed reduction, let us start with a high level description. We need to switch the u'th output ciphertext from real to an encryption of 0^k. First, let us switch how the u'th output ciphertext is generated, by always choosing the codeword symbols c_j in positions $j \in S^*$ uniformly at random and then choosing the remaining codeword symbols via the Extend procedure to ensure that the codeword encodes the intended plaintext – by definition, this yields an identically distributed codeword.

Second, let vk^* be the verification key in the u'th output ciphertext. We switch how all the oracle queries are answered: instead of decrypting with the real secret key, we will decrypt each input ciphertext that has verification key vk' using some row i in which the i'th bit of vk^* and vk' differ. By Fact 7, if the bad event does not occur before the v'th query, then all the oracle queries are answered identically independent of which row i is used, and therefore this change cannot affect the probability of the event $\mathsf{1stBad}_v$ occurring. With the above changes, the oracle queries are answered without any knowledge the component secret keys $\mathsf{dk}_{i,j,\mathsf{vk}_i^*}$ for $i \in [\kappa], j \notin S^*$. Intuitively this lets us replace the encrypted value in the u'th output ciphertext from real to an encryption of 0^k, since the only component ciphertexts that depend on the plaintext are those encrypted under component public keys $\mathsf{ek}_{i,j,\mathsf{vk}_i^*}$ for $i \in [\kappa], j \notin S^*$, for which the secret keys are no longer used by the oracle. However, although these secret keys are not used by the oracle, they are needed to check if the event $\mathsf{1stBad}_v$ occurred, since this depends on all the components of all input ciphertexts in oracle queries $1, \ldots, v$. The key insight is that we can rely on non-malleability security of the component scheme to check if the event $\mathsf{1stBad}_v$ occurred by making one parallel decryption query on all these ciphertexts at the very end of the game.

To make the above formal, fix some $u \leq v \in [q]$, and we describe a tag-nmCPA attacker against the component encryption $\mathcal{E}$, whose advantage is equal to

$$\left| \Pr_{H_{u-1}|_v}[\mathsf{1stBad}_v] - \Pr_{H_u|_v}[\mathsf{1stBad}_v] \right| = \left| \Pr_{H_{u-1}}[\mathsf{1stBad}_v] - \Pr_{H_u}[\mathsf{1stBad}_v] \right|$$

The reduction. The tag-nmCPA attacker, denoted B, begins by choosing at random a signature key-pair $(\mathsf{sk}^*, \mathsf{vk}^*) \leftarrow \mathrm{Gen}_S(1^\lambda)$ and a size-t subset $S^* \subset [n]$. It also chooses at random $\kappa(n+t)$ key-pairs for the component encryption scheme, setting $(\mathsf{dk}_{i,j,b}, \mathsf{ek}_{i,j,b}) \leftarrow \mathrm{Gen}_E(1^\lambda)$ for every $i \in [\kappa], j \in [n]$ and $b = \overline{\mathsf{vk}_i^*}$, and also for every $i \in [\kappa], j \in S^*$ and $b = \mathsf{vk}_i^*$. In words, B chooses one of every pair of keys (i,j,b) for $j \notin S^*$ (corresponding to the bit $b = \overline{\mathsf{vk}_i^*}$), and both keys $(i,j,0), (i,j,1)$ for $j \in S^*$.

B then receives $\kappa(n-t)$ public keys from its challenger, and assigns them to the missing positions (i,j,b) for $j \notin S^*$ and $b = \mathsf{vk}_i^*$. This completes the public key for the compound scheme, $\mathsf{ek}' = \{\mathsf{ek}_{i,j,b} : b \in \{0,1\}, i \in [\kappa], j \in [n], \}$, which B sends to the 1-FuncCPA$^+$ attacker A.

We note that component secret keys that B knows allow it to decrypt all input ciphertext queries, *except those with verification key* vk^*. Indeed for every other verification key $\mathsf{vk} \neq \mathsf{vk}^*$ there is at least one index i^* with $\mathsf{vk}_{i^*} \neq \mathsf{vk}_{i^*}^*$, and therefore B has a full functioning compound secret key

$$\mathsf{dk}' = \left(i^*, S^*, \{\mathsf{dk}_{i,j,b} : b \in \{0,1\}, i = i^* \text{ or } j \in S^*\}\right),$$

that it can use to decrypt. (Moreover, by Fact 7, if Bad_v doesn't happen then A will not be able to tell which row was used to answer that query.)

Next, B needs to answer the functional re-encryption queries that A makes. Let (ct'_k, f_k) be the k'th functional re-encryption query of A. We can assume that the verification keys in all the input ciphertexts ct'_k are different from vk^* since (a) for $k \leq u$ A has no information yet on vk^* and therefore $\mathsf{vk}_u \neq \mathsf{vk}^*$ except with a negligible probability, and (b) for $k > u$ we get $\mathsf{vk}_k \neq \mathsf{vk}^*$ since A is a conforming adversary. Hence, by the observation above B can decrypt all these queries. Let pt_k be the plaintext that B decrypts for the k'th query. B replies to the functional re-encryption queries as follows:

- For $k = 1, 2, \ldots, u-1$, B answer these queries simply by encrypting the all-zero plaintext.
- For $k = u+1, \ldots, v$, B replies to the k'th query by encrypting $f_k(\mathsf{pt}_k)$.
- For $k = u$, B uses its challenge-ciphertext oracle for the component scheme: B first chooses at random some $y \leftarrow \Sigma^t$ and extends it to get separate encodings of both the all-zero plaintext as well as the plaintext $f_k(\mathsf{pt}_k)$. Namely it sets $c^{k,0} := \mathrm{Extend}(0^k, y, S^*)$ and $c^{k,1} := \mathrm{Extend}(f_k(\mathsf{pt}_k), y, S^*)$. (Note again that $c^{k,0}, c^{k,1}$ agree on all the columns in S^*, $c^{0,k}|_{S^*} = c^{1,k}|_{S^*} = y$.)

  B makes a call to its challenge-ciphertext oracle, relative to all the $\kappa(n-t)$ component public keys that it received from its challenger, specifying the words $(c_j^{0,k} : j \notin S^*)$ and $(c_j^{0,k} : j \notin S^*)$ for each "row" i of public keys. It receives back the ciphertexts $\{\mathsf{ct}_{i,j,b} : i \in [\kappa], j \notin S^*, b = \mathsf{vk}^*_i\}$, encrypting one of these two words in all the rows. B extends them to a full compound ciphertext by encrypting the symbols in y for the columns in S^* (which are the same between $c^{k,0}$ and $c^{k,1}$), relative to the appropriate public keys $\mathsf{ek}_{i,j,b}$ for all $i \in [\kappa], j \in S^*$ and $b = \mathsf{vk}^*_i$.

Concatenating all these component ciphertexts to a vector $\vec{\mathsf{ct}}$, B uses the signing key vk^* to generate a signature σ^*, and replies to A with the output ciphertext $\mathsf{ct}'_u = (\vec{\mathsf{ct}}, \mathsf{vk}^*, \sigma^*)$.

Depending on the answer from the challenge-ciphertext oracle of B, this is indeed a valid encryption of either the all-zero plaintext or the plaintext $f_k(\mathsf{pt}_k)$. Moreover, since Extend$(\cdots)$ yields the same distribution on codewords as $\mathsf{E}(\cdots)$, then we also get the right distribution for this output ciphertext.

Finally, after A makes its v'th functional re-encryption query, B uses its parallel decryption query to determine if the event $\mathsf{1stBad}_v$ happened. This decryption query includes all the component ciphertexts in all the functional re-encryption queries $k = 1, 2, \ldots, v$, for which B is missing the component secret key.

Importantly, the tags in all these queries are different than the tag vk^* that B used for the query to its challenge-ciphertext oracle, hence this is a valid decryption query that B is allowed to make. Also, we note that a single decryption query at the end is sufficient, B does not need to make adaptive queries. Given these decryptions, B can determine if the event $\mathsf{1stBad}_v$ occurred or not, outputting 1 if it occurred and 0 if not.

Analysis of the reduction. Denote by H'_{u-1} the reduction experimenter where B's challenge-ciphertext oracle encrypts the encoded all-zero plaintext, and by H'_u the reduction where the oracle encrypts the encoded $f_u(\mathsf{pt}_u)$. Note that the only difference between H'_{u-1} and the hybrid H_{u-1} is that in H_{u-1} the same row is used to decrypt all the queries, whereas B uses different rows for different queries in H'_{u-1} (and the same holds for H'_u vs. H_u). [11]

However, due to Fact 7, as long as none of the bad events $\mathsf{Bad}_1, \ldots, \mathsf{Bad}_{v-1}$ happen, the view of A is independent of the row that was used to decrypt. And as soon as any of these events happen, we are ensured that $\mathsf{1stBad}_v$ does not happen (in any of H'_{u-1}, H_{u-1}, H'_u, and H_u). It follows that $\Pr_{H'_{u-1}}[\mathsf{1stBad}_v] = \Pr_{H_{u-1}}[\mathsf{1stBad}_v]$ and $\Pr_{H'_u}[\mathsf{1stBad}_v] = \Pr_{H_u}[\mathsf{1stBad}_v]$. Hence the advantage of B in the tag-nmCPA game is exactly

$$\left| \Pr_{H'_{u-1}}[\mathsf{1stBad}_v] - \Pr_{H'_u}[\mathsf{1stBad}_v] \right| = \left| \Pr_{H_{u-1}}[\mathsf{1stBad}_v] - \Pr_{H_u}[\mathsf{1stBad}_v] \right|,$$

as needed. This completes the proof of Lemma 3. □

3.4.3 Proof of Lemma 4

Lemma 4. *For all $v \in [q]$ there is a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr_{H_q}[\mathsf{Bad}_v] < \mathsf{negl}(\lambda)$.*

Proof. Recall that in the hybrid H_q, all functional re-encryption queries are answered with a fresh encryption of the all-zero plaintext, regardless of the input ciphertext. Hence, the view of A in that hybrid is independent of the set S^* of columns that is used to check the ciphertext during decryption. We can therefore analyze the probability of the event Bad_v in a modified game, in which the set S^* is chosen at random after the v'th decryption query.

Let $\mathsf{ct}' = (\vec{\mathsf{ct}}, \mathsf{vk}, \sigma)$ be the input ciphertext in the v'th query, and denote by $c_i \in \{0, 1, \perp\}^n$ the word obtained by decrypting the i'th ciphertext row. To trigger the bad event Bad_v, the first conditions says that ct' must be an invalid ciphertext, so there are indices $i_1, i_2 \in [\kappa]$, such that $\mathsf{D}'(c_{i_1}) = \bar{c} = (\bar{\gamma}_1, \ldots, \bar{\gamma}_n)$, but $c_{i_2} = (\gamma_{i_2,1}, \ldots, \gamma_{i_2,n})$ is at Hamming distance more than d from $\bar{c}$. But in that case the probability (over S^* chosen as a random size-t subset of $[n]$) of the second condition holding, $\gamma_{i_2,j} = \bar{\gamma}_j$ for all $j \in S^*$, is bounded by $(1 - \frac{d}{n})^t \leq 2^{-\lambda}$. □

[11] An "invisible" difference is that u'th output ciphertext is computed using Extend rather than applying the encoding $\mathsf{E}(\cdots)$, but this produces the same distribution over the codewords.

3.4.4 Proof of Lemma 5

Lemma 5. *For all $u \in [q]$ there is a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\left| \Pr_{H_u}[A \to 1 \ \& \ \overline{\mathsf{Bad}}] - \Pr_{H_{u-1}}[A \to 1 \ \& \ \overline{\mathsf{Bad}}] \right| \leq \mathsf{negl}(\lambda).$$

Proof. The proof is nearly identical to Lemma 3, by reduction to the tag-nmCPA-security of the component encryption scheme $\mathcal{E}$. The only differences are (a) the reduction continues all the way to the end of the game (rather than aborting it after the v'th functional re-encryption query), and (b) the reduction algorithm's output at the end is calculated differently.

Specifically, the tag-nmCPA attacker B begins exactly the same as in the proof of Lemma 3, and answers functional re-encryption queries of A in exactly the same way. Once A halts with some output bit b, the attacker B uses its parallel decryption query to determine if the event Bad happened. This decryption query includes all the component ciphertexts in all the functional re-encryption queries $k = 1, 2, \ldots, q$, for which B is missing the component secret key. (As before, the tags in all these queries are different than the tag vk^* that B used for the query to its challenge-ciphertext oracle, hence this is a valid decryption query that B is allowed to make.) Given these decryptions, B can determine if the event Bad occurred or not. B then outputs 1 if A *returned 1 and* Bad *did not occur*, and 0 otherwise.

As in the proof of Lemma 3, the view of A in the reduction is identical to its view in the hybrids H_{u-1} or H_u as long as the event Bad did not occur, and therefore the advantage of B is equal to $|\Pr_{H_u}[A \to 1 \ \& \ \overline{\mathsf{Bad}}] - \Pr_{H_{u-1}}[A \to 1 \ \& \ \overline{\mathsf{Bad}}]|$. □

Remark 1. *We note that since we applied the CDMW transformation to a non-malleable scheme (which is in particular CPA secure), then the resulting $\mathcal{E}'$ is also non-malleable, not just 1-FuncCPA$^+$ secure. In fact, it can simultaneously withstand any number of adaptive functional re-encryption queries and a single parallel decryption query at the end of the game.*

On the other hand, it is easy to see that this scheme is not CCA-secure (not even CCA1-secure). This is true for the same reason that the original CDMW transformation fails to produce a CCA-secure scheme: An attacker with adaptive access to a decryption oracle can use that oracle to detect the columns in the special subset S^*, then figure out the special row i^*, and then completely break the scheme.

3.5 Extension to General FuncCPA-Security

The proof of Lemma 2 extends easily to show multi-input FuncCPA$^+$ security, and not just 1-FuncCPA$^+$ security. In fact, the proof is essentially identical with minor syntactical modifications:

- The simplifying assumptions on the adversary remain the same, but now there are multiple input ciphertexts for each query. In partcicular, without loss of generality, we can assume that (a) *none* of the input ciphertexts in any query are equal to any previous output ciphertext, (b) the signatures of *all* of the input ciphertexts verify correctly, (c) the sets of verification keys in the input ciphertexts is disjoint from that in the output ciphertexts.

- We define the Bad event for a multi-input query is triggered if any of the ciphertexts in that query satisfy the original definition of the Bad event.

- The rest of the proof proceeds identically. In the proof of Lemma 4, we need to take an additional union bound over all ℓ input ciphertexts in the v'th query.

4 Conclusions and Open Problems

In this work we proved that FuncCPA secure encryption can be constructed from any CPA scheme, essentially by *applying twice* the CPA-to-mnCPA transformation of Choi *et al.* [CDMW18a]. A remaining open problem is to come up with simpler constructions, and in particular to resolve the question of whether a single application of this transformation suffices.

A similar question can be asked about non-functional re-encryption oracles, if it is easier to construct a secure scheme against non-functional re-encryption oracles from CPA than one secure against functional re-encryption? We remark that sometimes it is easier to withstand non-functional re-encryption. For example, ElGamal encryption is easily seen to be secure against non-functional re-encryption. However, we do not know if it can be proven secure against functional re-encryption under a reasonable assumption (or, conversely, if there is some surprising attack). More generally, it might be interesting to build natural number-theoretic FuncCPA$^+$-secure scheme which are not CCA1-secure.

Another open problem is the relation between FuncCPA and FuncCPA$^+$: we have shown implication in one direction (and separation for non-functional re-encryption oracles), but the other direction for functional re-encryption oracles remains open. If the general separation is found, it would be interesting to see if there are any real-world applications which require the stronger form of FuncCPA$^+$ security, and could be insecure with FuncCPA security.

More generally, it would be good to find more applications of FuncCPA and FuncCPA$^+$ encryption schemes.

References

[AGHV22] Adi Akavia, Craig Gentry, Shai Halevi, and Margarita Vald. Achievable CCA2 relaxation for homomorphic encryption. In Eike Kiltz and Vinod Vaikuntanathan, editors, *Theory of Cryptography - 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part II*, volume 13748 of *Lecture Notes in Computer Science*, pages 70–99. Springer, 2022. Also available from https://ia.cr/2022/282. 1, 4, 20

[AV22] Adi Akavia and Margarita Vald. private communication, Nov. 2022. 4

[CDM$^+$20] Sandro Coretti, Yevgeniy Dodis, Ueli Maurer, Björn Tackmann, and Daniele Venturi. Non-malleable encryption: Simpler, shorter, stronger. *J. Cryptol.*, 33(4):1984–2033, 2020. 2, 9, 10

[CDMW18a] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. A black-box construction of non-malleable encryption from semantically secure encryption. *J. Cryptol.*, 31(1):172–201, 2018. 2, 3, 8, 17

[CDMW18b] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Improved, black-box, non-malleable encryption from semantic security. *Des. Codes Cryptogr.*, 86(3):641–663, 2018. 2, 3

[CHH$^+$07] Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Bounded cca2-secure encryption. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, volume 4833 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2007. 2

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC '99, Kamakura, Japan, March 1-3, 1999, Proceedings*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68. Springer, 1999. 2

[HKW20] Susan Hohenberger, Venkata Koppula, and Brent Waters. Chosen ciphertext security from injective trapdoor functions. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 836–866. Springer, 2020. 2

[MSS13] Steven A. Myers, Mona Sergi, and Abhi Shelat. Black-box construction of a more than non-malleable CCA1 encryption scheme from plaintext awareness. *J. Comput. Secur.*, 21(5):721–748, 2013. 2

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 427–437. ACM, 1990. 2

[PSV06] Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 271–289. Springer, 2006. 2, 6, 8, 9

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 543–553. IEEE Computer Society, 1999. 2

A Direct Implications and Separations

It follows directly from the definitions above that every CCA2-secure scheme is also nmCPA-secure, which is in turn also CPA secure. Additionally, in Lemma 1 we prove that every FuncCPA$^+$-secure scheme is also FuncCPA-secure, which is in turn CPA secure by definition. Here we study the other (non-)implications.

First, while our intuition tells us that every CCA2-secure scheme should also be FuncCPA$^+$ secure, we note that this implication is not completely straightforward, because the FuncCPA attacker is allowed to copy the challenge ciphertext for its re-encryption queries, while the CCA2-attacker is not allowed to do so. Nonetheless, we show that our intuition is still correct. In fact, we show that already (weaker) CCA1-security implies FuncCPA$^+$ security.

Lemma 8. *Every CCA1-secure encryption scheme is also FuncCPA$^+$-secure. (In particular, CCA2-security implies the original FuncCPA-security.)*

Proof. Let q be the overall number of re-encryption queries made by the FuncCPA$^+$-attacker A. For $0 < i \leq q$, we define hybrid H_i where the first i re-encryption queries $(\vec{\mathsf{ct}}, f)$ by A return $\mathsf{E}(\mathsf{ek}, f(\mathrm{Dec}(\mathsf{dk}, \vec{\mathsf{ct}})))$, and the remaining $(q-i)$ queries return $\mathsf{E}(\mathsf{ek}, 0)$. By hybrid argument it is enough to prove that H_i is indistinguishable from H_{i+1}, for every $0 < i \leq q$, as H_0 and H_q corresponding to $b = 0$ and $b = 1$ experiments, respectively.

For the latter, we have the following almost immediate reduction to CCA1-security from Definition 2. To simulate the j-th query (ct_j, f_j) of an attacker A claiming to distinguish H_i from H_{i+1}, the CCA1 attacker B does the following:

- For $j < i$, query its decryption oracle $\mathcal{O}_1$ on the ciphertexts in $\vec{\mathsf{ct}}_j$, obtaining plaintexts $\vec{\mathsf{pt}}_j$. Compute $\mathsf{pt}'_j = f_j(\vec{\mathsf{pt}}_j)$, and return to A an honestly generated $\mathsf{ct}'_j = \mathrm{Enc}(\mathsf{ek}, \mathsf{pt}'_j)$.
- For $j = i$, query its decryption oracle $\mathcal{O}_1$ on the ciphertexts in $\vec{\mathsf{ct}}_i$, obtaining plaintexts $\vec{\mathsf{pt}}_i$. Compute $\mathsf{pt}^* = f_i(\vec{\mathsf{pt}}_i)$, and submit the tuple $(\mathsf{pt}^*, 0)$ as its challenge.
  Finally, return to A the resulting challenge ciphertext ct^* to the attacker.
- For $j > i$, ignore $(\vec{\mathsf{ct}}_j, f_j)$, and return $\mathsf{E}(\mathsf{ek}, 0)$ to A.

For $b = 0$, this run of B is a perfect simulation of H_i, while for $b = 1$ it is a perfect simulation of H_{i+1}, completing the proof. □

In the opposite direction, Akavia et al. demonstrated in [AGHV22] that (somewhat surprisingly) CPA security of a scheme *does not* imply even the most basic ReEncCPA security of the same scheme. Below we extend their example to show that non-malleability (i.e., nmCPA-security) of a scheme also does not imply even ReEncCPA security. We also demonstrate that even FuncCPA$^+$ security does not imply nmCPA-security.

Lemma 9. *If nmCPA-secure encryption schemes exist, then there exists a nmCPA-secure encryption scheme which is not ReEncCPA-secure. Conversely, if FuncCPA$^+$-secure encryption schemes exist, then there exists a FuncCPA$^+$-secure encryption scheme which is not nmCPA-secure.*

Proof. Starting from the easy separation, we can append 0 to all honestly produced ciphertexts in a FuncCPA$^+$-secure encryption scheme, and have the decryption oracle simply ignore this appended bit. This clearly does not change FuncCPA$^+$-security, as all honestly re-encrypted ciphertexts will still end with 0. However, the scheme is obviously malleable, by flipping the last bit of the challenge ciphertext from 0 to 1, and calling the decryption oracle of the resulting (formally "distinct") ciphertext.

For the other separation, let $\mathcal{E} = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ be a scheme which is nmCPA-secure according to definition 2, and we modify it into a scheme $\mathcal{E}' = (\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}')$ as follows:

- Gen' just runs Gen twice, outputting the two pairs $((\mathsf{dk}, \mathsf{dk}'), (\mathsf{ek}, \mathsf{ek}'))$. Roughly, dk, ek are the "real keys" for decryption and encryption, whereas $\mathsf{dk}', \mathsf{ek}'$ are used for signalling various events.
- The new encryption $\mathrm{Enc}'((\mathsf{ek}, \mathsf{ek}'), \mathsf{pt})$ checks if pt is the secret key corresponding to either ek or ek':
  - If pt is the secret key corresponding to ek or ek' then output $1|\mathsf{pt}$,
  - Otherwise output $0|\mathrm{Enc}(\mathsf{ek}, \mathsf{pt})$.
- The new decryption $\mathrm{Dec}((\mathsf{dk}, \mathsf{dk}'), \mathsf{ct}')$ parses $\mathsf{ct}' = b|\mathsf{ct}$ with $b \in \{0, 1\}$, then proceeds as follows:
  - If $b = 1$ and $\mathsf{ct} = \mathsf{dk}'$ then output dk,
  - If $b = 1$ and $\mathsf{ct} \neq \mathsf{dk}'$ then output dk',
  - Otherwise output $\mathrm{Dec}(\mathsf{dk}, \mathsf{ct})$.

It is easy to see that the modified $\mathcal{E}'$ is still nmCPA-secure: An nmCPA attack on $\mathcal{E}'$ can be turned into nmCPA attack on the underlying $\mathcal{E}$ by having the reduction generate $(\mathsf{dk}', \mathsf{ek}')$ itself, then simulate the sole decryption query to $\mathcal{E}'$ using its decryption oracle to $\mathcal{E}$: Unless the $\mathcal{E}'$ attacker guesses dk' (on which it has no information other than seeing ek'), then it cannot trigger the 1st bullet on decryption above.

On the other hand, it is easy to see that a ReEncCPA attacker can break this scheme completely, first making a query with $\mathsf{ct} = 11\ldots1$ to get $1|\mathsf{dk}'$, then making a second query with $1|\mathsf{dk}'$ to get "the real key" dk. □

Next, we show separation between ReEncCPA and ReEncCPA$^+$ notions (and conjecture that similar separations hold for FuncCPA and 1-FuncCPA notions).

Lemma 10. *If ReEncCPA-secure encryption schemes exist, then there exists a ReEncCPA-secure encryption scheme which is not ReEncCPA$^+$-secure.*

Proof. Let $\mathcal{E} = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ be a scheme which is ReEncCPA-secure according to definition 2, and we modify it into a scheme $\mathcal{E}' = (\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}')$ as follows: The key generation remains unchanged, $\mathrm{Gen}' = \mathrm{Gen}$. Encryption is modified by setting

$$\mathrm{Enc}'(\mathsf{ek}, \mathsf{pt}) = \begin{cases} 11\ldots1 & \text{if } \mathsf{pt} \text{ is a decryption key corresponding to } \mathsf{ek} \\ 0|\mathrm{Enc}(\mathsf{ek}, \mathsf{pt}) & \text{otherwise.} \end{cases}$$

(Note that it is possible to check efficiently whether the condition above holds.) Decryption is also modified, as follows:

$$\mathrm{Dec}'(\mathsf{dk},\mathsf{ct}') = \begin{cases} \mathsf{dk} & \text{if } \mathsf{ct}' \text{ begins with a } 1 \\ \mathrm{Dec}(\mathsf{dk},\mathsf{ct}) & \text{if } \mathsf{ct}' = 0|\mathsf{ct}. \end{cases}$$

It is easy to see that $\mathcal{E}'$ is still ReEncCPA-secure according to definition 2 (with a non-functional decryption oracle), since access to the oracle for $\mathcal{E}'$ can be perfectly simulated using access to the oracle for $\mathcal{E}$. (Indeed ciphertext beginning with 1 are answered with $11\ldots1$ and ciphertexts beginning with 0 are answered as in $\mathcal{E}$, with a zero prepended to the reply.) On the other hand, it is easy to distinguish a true re-encryption oracle from a zero-encrypting one, just by querying it on any ciphertext that begins with a 1. □

Finally, we show that a 1-FuncCPA$^+$-secure scheme is not necessarily FuncCPA-secure (and, thus, not necessarily FuncCPA$^+$-secure), assuming the existence of CCA-secure schemes.

Lemma 11. *If CCA-secure encryption schemes exist, then there exists a* 1*-FuncCPA*$^+$*-secure encryption scheme which is not FuncCPA-secure.*

Proof. Let $\mathcal{E} = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ be a CCA-secure scheme, and let $OWF(\cdot)$ be a one-way function. (Recall that CCA-secure encryption implies the existence of one-way functions.) Consider the modified scheme $\mathcal{E}' = (\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}')$, defined as follows:

- $\mathrm{Gen}'(1^\lambda)$ runs the underlying key-generation $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathrm{Gen}(1^\lambda)$, and in addition chooses two uniformly random and independent strings $r, s \leftarrow \{0,1\}^\lambda$ and sets $y = OWF(r \oplus s)$. The public key is $\mathsf{ek}' = (\mathsf{ek}, y)$ and the secret key is $\mathsf{dk}' = (\mathsf{dk}, r, s)$.
- $\mathrm{Enc}'(\mathsf{ek}', \mathsf{pt})$: If $y = OWF(\mathsf{pt})$ then output pt, else output $(0, \mathrm{Enc}(\mathsf{ek}, \mathsf{pt}))$.
- $\mathrm{Dec}'(\mathsf{dk}', (b, \mathsf{ct}))$: If $b = 0$ then output $\mathrm{Dec}(\mathsf{dk}, \mathsf{ct})$. If $b = 1$ then output r, if $b = 2$ then output s.

We show that $\mathcal{E}'$ is 1-FuncCPA$^+$-secure, but not FuncCPA-secure. To see that $\mathcal{E}'$ is 1-FuncCPA$^+$-secure, let us again consider only adversaries that never use the answers from previous re-encryption queries as inputs to future queries. (As we argued before, we can make this assumption without loss of generality.) Fixing one such adversary, we consider a sequence of hybrids, where in the i'th hybrid the first $i-1$ queries are answered by encryption of 0, and the i'th query and later are answered by the single-ciphertext re-encryption oracle. Arguing that hybrid i is indistinguishable from hybrid $i+1$ is done in two steps:

- We first argue that the i'th query will not decrypt to $r \oplus s$ (except with a negligible probability), by reduction to the one-wayness of $OWF(\cdot)$. Here, the reduction algorithm is given the secret key dk of the underlying encryption scheme Enc.
- Then we replace the i'th query answer by an encryption of zero, and argue indistinguishability by reduction to the CCA-security of the underlying scheme $\mathcal{E}$. Here the reduction algorithm is given access to the decryption oracle of $\mathcal{E}$, that allows it to simulate the answers to all future queries.

On the other hand, it is clear that $\mathcal{E}'$ is *not FuncCPA-secure.* The multi-ciphertext re-encryption oracle is easily distinguishable from a zero-encrypting oracle, because it enables easy extraction of a pre-image of y under $OWF(\cdot)$: The multi-ciphertext query $(\mathsf{ct}_1 = (1, 0^\lambda), \mathsf{ct}_2 = (2, 0^\lambda), f = \oplus)$ will decrypt ct_1 to r and ct_2 to s, then compute $x = f(r,s) = r \oplus s$, and applying the modified encryption procedure it will return the pre-image x. (As above, obtaining a pre-image of y is is hard given a zero-encrypting oracle, by reduction to the one-wayness of $OWF(\cdot)$.) □