Public-Key Anamorphism in (CCA-secure) Public-Key Encryption and Beyond

Giuseppe Persiano* Duong Hieu Phan† Moti Yung‡

August 24, 2024

Abstract: The notion of (Receiver-) Anamorphic Encryption was put forth recently to show that a dictator (i.e., an overreaching government), which demands to get the receiver's private key and even dictates messages to the sender, cannot prevent the receiver from getting an additional covert anamorphic message from a sender. The model required an initial private collaboration to share some secret. There may be settings though where an initial collaboration may be impossible or performance-wise prohibitive, or cases when we need an immediate message to be sent without private key generation (e.g., by any casual sender in need). This situation, to date, somewhat limits the applicability of anamorphic encryption.

To overcome this, in this work, we put forth the new notion of "*public-key anamorphic encryption*," where, without any initialization, any sender that has not coordinated in any shape or form with the receiver, can nevertheless, under the dictator control of the receiver's private key, send the receiver an additional anamorphic secret message hidden from the dictator. We define the new notion with its unique new properties, and then prove that, quite interestingly, the known CCA-secure Koppula-Waters (KW) system is, in fact, public-key anamorphic.

We then describe how a public-key anamorphic scheme can support a new *hybrid anamorphic* encapsulation mode (KDEM) where the public-key anamorphic part serves a bootstrapping mechanism to activate regular anamorphic messages in the same ciphertext, thus together increasing the anamorphic channel capacity.

Looking at the state of research thus far, we observe that the initial system (Eurocrypt'22) that was shown to have regular anamorphic properties is the CCA-secure Naor-Yung (and other related schemes). Here we identify that the KW CCA-secure scheme also provides a new type of anamorphism. Thus, this situation is hinting that there may be a connection between some types of CCA-secure schemes and some type of anamorphic schemes (in spite of the fact that the goals of the two primitives are fundamentally different); this question is foundational in nature. Given this, we identify a sufficient condition for a "CCA-secure scheme which is black-box reduced from a CPA secure scheme" to directly give rise to an "anamorphic encryption scheme!" Furthermore, we identify one extra property of the reduction, that yields a public-key anamorphic scheme as defined here.

*Università di Salerno, Italy and Google LLC, USA. giuper@gmail.com

†Telecom Paris, Institut Polytechnique de Paris, France. hieu.phan@telecom-paris.fr

‡Google LLC and Columbia University, USA. motiyung@gmail.com

Contents

1 Introduction and Motivation **3**
1.1 Our contributions 5

2 Definitions **7**
2.1 Anamorphic Encryption Schemes 9
2.2 Public-Key Anamorphic Encryption scheme 9

3 Public-Key Anamorphic Scheme: The KW construction **10**
3.1 Hinting PRF 11
3.2 The encryption scheme 12
3.3 KW is anamorphic 14

4 Anamorphic Hybrid Encryption: KW as a KDEM **15**

5 CCA security and anamorphic encryption **17**
5.1 The intuition 18
5.2 Reductions that give Public Anamorphism 19
5.3 Security Games and Reductions 20
5.4 The reduction from the KW proof 24
5.5 Sufficient Conditions for Anamorphism 24
5.6 The Anamorphic Triplet 27

6 Conclusions **28**

A A Review of the Naor-Yung Encryption Scheme **32**

1 Introduction and Motivation

Receiver-Anamorphic Encryption[1] has been introduced in [PPY22] to provide private end-to-end communication even in the presence of a dictator that requests and gets to see the secret key associated with the public key (modeling an over reaching by government/ authority engaged in the crypto wars). The naming of the strong adversary "a dictator" should not be taken literally, the work on anamorphism, in fact, is a structural work which attempts to understand and implement re-purposing of cryptosystems (needed under certain conditions). Roughly speaking, an asymmetric encryption scheme $\mathsf{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is an *anamorphic* encryption scheme if there exists a triplet of *anamorphic* algorithms $\mathsf{T} = (\mathsf{aKG}, \mathsf{aEnc}, \mathsf{aDec})$ that allows to embed *anamorphic messages* into ciphertexts carrying regular messages. More specifically, the *anamorphic key generation* algorithm aKG outputs a pair of public and secret keys $(\mathtt{pk}, \mathtt{sk})$ along with a *double key* $\mathtt{dkey}$. The double key $\mathtt{dkey}$ is used by the *anamorphic encryption* algorithm to embed an anamorphic message $\mathtt{amsg}$ in a ciphertext $\mathtt{ct}$ carrying a regular message $\mathtt{msg}$. The double key $\mathtt{dkey}$ is also used by the *anamorphic decryption* algorithm to extract $\mathtt{amsg}$ from $\mathtt{ct}$.

The concept of an anamorphic encryption scheme is designed with the purpose of regaining privacy in the presence of an adversary (called dictator) that requests access to the secret key $\mathtt{sk}$ in the following way. Bob sets up a public key for encryption scheme E but, instead of using KG, he runs the anamorphic algorithm aKG that returns $(\mathtt{pk}, \mathtt{sk}, \mathtt{dkey})$. The public key $\mathtt{pk}$ is made public, the decryption key $\mathtt{sk}$ is kept secret and the double key $\mathtt{dkey}$ is shared with Alice. If the dictator asks for the secret key, Bob releases $\mathtt{sk}$. Note that Bob has ostensibly set up his key by running the key generation algorithm KG of E and thus he can plausibly deny that he has no double key. Whenever Alice wants to send a message $\mathtt{amsg}$ that must be kept secret from the dictator, Alice picks an innocent looking regular message $\mathtt{msg}$ and uses aEnc on input $\mathtt{msg}, \mathtt{amsg}$, and $\mathtt{dkey}$. The ciphertext $\mathtt{ct}$ produced by aEnc has the following anamorphic property: if $\mathtt{ct}$ is decrypted by Enc using $\mathtt{sk}$, then $\mathtt{msg}$ is given as output; if instead $\mathtt{ct}$ is decrypted by aEnc using $\mathtt{dkey}$, then $\mathtt{amsg}$ is returned. In other words, the dictator that has $\mathtt{sk}$ will be able to extract the innocent regular message $\mathtt{msg}$ whereas Bob will obtain $\mathtt{msg}$ and $\mathtt{amsg}$.

Obviously, for the Dictator to be fooled, the use of the anamorphic triplet T must be indistinguishable from the use of E even if the secret key $\mathtt{sk}$ is available. This means that KG and aKG must induce indistinguishable distributions of the pair $(\mathtt{pk}, \mathtt{sk})$ and Enc and aEnc must produce indistinguishable ciphertexts. This is captured by the formal notion of anamorphic encryption scheme put forth in [PPY22] and used also in [KPP$^+$23b].

As observed in [PPY22], designing new and contrived anamorphic encryption schemes is a rather easy goal which produces suspicious systems. In contrast, one should rather look at existing encryption schemes (which were *not* designed with anamorphism in mind) and design anamorphic triplets for them (i.e. re-purposing a scheme). Indeed, Bob should be able to plausibly deny the existence of a double key which can be natural if he declares to deploy a well known encryption scheme. On the other hand, setting up the public key of a scheme that is designed to be anamorphic would look rather suspicious and will be self accusatory. Rather surprisingly, recent works on anamorphic encryption [PPY22, KPP$^+$23b] have established the prevalence of anamorphism in currently available encryption schemes.

[1] In this paper we will only discuss Receiver-Anamorphic Encryption and therefore we will drop *Receiver* and only talk of *Anamorphic Encryption*.

Settings where Previous Constructions are Limiting. In this work we point at some cases where there are constraints that limit the usability of the prior works [PPY22, KPP$^+$23b, BGH$^+$24, WCHY23]. Indeed, the current examples of anamorphic encryption do provide a private channel from Alice to Bob even if the Dictator has Bob's secret key, but this comes at the expense of certain *operational* cost and of some *technical* issues.

From the *operational* point of view, we see that the normal encryption scheme is an asymmetric scheme whereas the anamorphic encryption scheme is symmetric. This is due to the fact that the anamorphic encryption and decryption requires information, namely the double key, that must be kept hidden from the Dictator. Since it must be used for both encryption and decryption, the double key must be shared using a private channel (or some covert key exchange), thus greatly affecting the operating conditions of the anamorphic scheme. In other words, an asymmetric encryption scheme for the regular message technically downgrades to symmetric when going to the counterpart anamorphic message. Clearly, this limits the application of the concept to parties that have previously interacted. (The limitation further exacerbates when many senders want to use the anamorphic channel!) We thus ask:

> *is the limiting operating conditions caused by the downgrade from asymmetric to symmetric encryption necessary? Or is it possible to preserve the asymmetric nature of the regular encryption scheme when moving to the anamorphic message sending in the anamorphic setting?*

The literature presents two steps in this direction but we do not have a satisfactory answer to the question above, yet. First, we mention the notion sender-anamorphic of [PPY22, WCHY23]. This notion can be indeed achieved without initially sharing a key (but in a different adversary which dictates message to the sender). Further, the cost of this is two additional assumptions: (1) the sender knows, both, the forced receiving public key (this is imposed by the dictator) and the duplicate existing receiving public key; (2) the latter must be kept hidden from the dictator just like the double key. Secondly, Banfi et al. [BGH$^+$24] instead decouple the generation of the double key from the generation of the public and secret key. This allows anamorphic encryption to be used for existing legacy public keys but anamorphism still relies on the secrecy of a double key.

From the *technical* point of view, instead, we see a degradation of the cryptographic security guarantees offered by the encryption scheme with respect to parties that have access to the double key. Specifically, the CCA-secure encryption schemes based on Smooth Projective Hash Functions of [KPP$^+$23b] lose CPA-security with respect to users that have access to the double key (considered collaborators and non-attackers). In other words, all users that have access to the double key can read the regular message as well as the anamorphic message, if any, embedded in a ciphertext (and therefore are assumed to be trusted in this respect). The Cramer-Shoup (proved to be anamorphic in [KPP$^+$23b]) and the Naor-Yung (proved to be anamorphic in [PPY22]) as wells as all CCA-secure encryption schemes based on NIZK do preserve CPA-security but they lose CCA-security with respect to parties that have access to the double key. In particular, this means that the double key may (in principle) allow to maul ciphertexts. Given this situation, we ask:

> *is this security downgrade necessary? Can we have an anamorphic CCA-secure encryption scheme that remains CCA-secure even with respect to parties that are allowed to send anamorphic messages?*

1.1 Our contributions

Let us review our conceptual and technical results.

Public-key anamorphism: a new notion. The above discussion motivates us to introduce a new model for anamorphic encryption which does not suffer from the above issues, and to investigate it further. Specifically we introduce the concept of a *public-key anamorphic* encryption scheme. In this new notion, Bob can deploy the anamorphic encryption scheme E regularly or anamorphically. In the latter case, Bob obtains a double key $\mathtt{dkey}$ (by looking ahead, this will be called the receiver double key $\mathtt{rdk}$ in the formal definition). When Alice wants to send Bob a regular message $\mathtt{msg}$, she just uses the regular encryption algorithm Enc on input Bob's public key. So far, similar to the previous case. When Alice wants to send an anamorphic message $\mathtt{amsg}$ and a regular message $\mathtt{msg}$ (for the dictator's use) then she runs the anamorphic encryption algorithm aEnc on input the two messages and Bob's *public key*. Note that there is *no* secret information that must be used by Alice but only a different (and publicly known) encryption algorithm. Note that aEnc will work also if the public key has been regularly generated by Bob and in this case the anamorphic message will remain forever inaccessible to Bob (and to the Dictator). When the ciphertext produced by Alice reaches Bob, it can be decrypted by using Dec and Bob's secret key thus recovering $\mathtt{msg}$. What happens to the anamorphic message $\mathtt{amsg}$. At first glance, it seems very counterintuitive that Alice could send an anamorphic message to Bob without agreeing on any secret shared information and relying only on Bob's public key. The subtle point is actually on Bob's side: if the public key has been set up anamorphically, then Bob can feed the ciphertext to aDec along with the (receiver) double key to obtain $\mathtt{amsg}$. We remark that no private channel is needed between Alice and Bob. This means that we have eliminated one possible security threat (what if the dictator eavesdrops on the private channel?) and there is no need for previous interaction between Alice and Bob. This is particularly useful if, say, Bob sets up his key to obtain messages from citizens that might be in distress because of the Dictator. In other words: This notion solves the operational drawback of the notion of anamorphic encryption as was put forth in [PPY22] by dispensing with the need of previous interaction to share information that must be kept hidden from the dictator.

We give a formal definition of a *public-key anamorphic* encryption scheme in Section 2.2. In Section 3, we show that the CCA-secure encryption scheme by Koppula and Waters [KW19] is indeed public-key anamorphic thus showing that this upgraded notion is realized by an existing encryption scheme.

Observe also that in the framework of public-key anamorphism, the technical drawback disappears as well. Indeed, Alice receives no further information regarding Bob's public key and thus, quite trivially, Alice has no advantage in breaking CCA security with respect to other users.

Public-key anamorphism as a KDEM. Public-key anamorphic encryption plays the same role as public-key encryption in the realm of traditional encryption. Indeed, before the introduction of public-key encryption, encryption was difficult to use because of the need of a secure channel to exchange a key beforehand. One of the primary uses of public-key encryption is to act as a *KEM* (key encapsulation mechanism) to privately and non-interactively exchange encryption keys for symmetric encryption. The symmetric encryption acts as a *DEM* (data encapsulation mechanism) and it is used to encrypt the actual data. This gives rise to the so called *hybrid encryption* mode where we have one asymmetric ciphertext acting as a KEM and one symmetric ciphertext acting as DEM. Public-key anamorphic encryption can do the same: as an anamorphic KEM for the

double key to be used with any other anamorphic encryption scheme that plays the role of a DEM. We will actually show that a public-key anamorphic encryption gives rise to a novel and more efficient encapsulation mechanism. In Section 4, we show, in particular, that the KW encryption scheme [KW19] has the property of acting as a *key-data encapsulation mechanism* (a *KDEM*), where *one single* ciphertext can encapsulate the double key (thus functioning as a KEM) *and simultaneously* securely carry the anamorphic messages (thus functioning as a DEM, as well). This development demonstrates that anamorphism may also enable new/enhanced functionalities.

Implications: direct relationships between CCA secure systems and anamorphic schemes. We note that there seems to be an underground link between CCA security and anamorphism. The first encryption scheme showed to be anamorphic is the CCA-secure encryption scheme by Naor and Yung [NY90] and since then other CCA-secure encryption schemes have been proved to be anamorphic both in the standard model (the Cramer-Shoup encryption scheme and the general construction based on Smooth Projective Hash Functions where proved anamorphic in [KPP$^+$23b]) and in the Random Oracle model (RSA-OAEP proved anamorphic in [KPP$^+$23b]). Moreover, the Canetti-Halevi-Katz construction of CCA-secure encryption from Identity based Encryption [CHK04] was proved secure in [KPP$^+$23a]. And, finally, in this paper we show that the first implementation of a new and stronger notion of anamorphism is actually achieved by a CCA-secure encryption scheme. This state of affairs is intriguing, hence in this paper we embark on the study of the deep connection between CCA-security and anamorphism.

Our main result is a set of sufficient conditions on the reduction used to prove CCA-security from CPA-security that guarantees that the CCA-secure encryption scheme is actually anamorphic. See Theorem 3 in Section 5. The sufficient conditions are general enough to be satisfied by the Koppula-Waters encryption scheme (shown to be public-key anamorphic) as well as by the NIZK-based constructions of CCA-secure encryption scheme as first put forth by Naor and Yung. By looking ahead we show that the step of the proof of CCA-security that relies on the CPA-security of an underlying scheme must contain a reduction that can be used to provide anamorphic encryption.

General conditions for public-key anamorphism. The acute reader might have noticed that not all CCA-secure encryption schemes give the same type of anamorphism even though they are derived through the same general technique relying on the CPA-security of an underlying scheme. As we shall see, this is due to the nature of the challenge ciphertext produced by the reduction. Reductions that produce *ambiguous* ciphertexts give rise to anamorphic encryption schemes that are not public-key. On the other hand, if the reduction produces non-ambiguous challenge ciphertexts then the resulting CCA-secure encryption schemes is indeed public-key anamorphic (see the notion of a public-key strong reduction in Definition 13). This is made formal in Theorem 4 (see Section 5) where we show that the KW public-key anamorphic construction is a special case of a general approach.

We point out that the general construction of anamorphism through the reduction is not meant to obtain new constructions (and indeed we do not) but mainly to illustrate the underpinning principles and reasons of why we see some CCA-secure schemes also giving support to anamorphism.

Roadmap. In Section 2, we put forth our new notion. We do so by first refining the notion of anamorphic encryption scheme so to point out which part of the double key is used by the sender

(denoted by sdk) and which by the receiver (denoted by rdk) and then defining the notion of public-key anamorphism by requiring $\mathtt{sdk} = \perp$.

In Section 3, we show that the Koppula-Water CCA-secure encryption scheme is indeed public-key anamorphic, whereas in Section 4 we describe KDEM, the new encapsulation mode of public-key anamorphic schemes. Finally, in Section 5, we give sufficient conditions for CCA-security to imply anamorphism.

More related work. As already discussed, the notion of anamorphic encryption has been introduced in [PPY22] and the prevalence of anamorphic encryption has been established by Kutylowski et al. [KPP+23b]. The notion of anamorphism has been extended from basic forms of encryption to homomorphic encryption by Catalano et al. [CGM24] and the concept of an anamorphic signature scheme has been introduced in [KPP+23a]. We stress again that the notion of sender-anamorphic of [PPY22] still requires the duplicate receiving public key to be hidden from the dictator. Indeed, this is used by the coin-toss faking algorithm $\mathsf{fRandom}$ and if the dictator gains knowledge of the receiving public key, then anamorphism is compromised.

Banfi et al. [BGH+24] have extended the original notion of anamorphic encryption in two important respects. First they present a more flexible notion of anamorphism in which the generation of anamorphic and regular keys are decoupled thus making it possible to add anamorphism to legacy keys. This greatly improves the deployability of the concept but it still relies on the double key to be hidden from the dictator. In addition, they introduce a natural robustness notion which states that the anamorphic decryption algorithm will not return any message when applied on a regular ciphertext. Fischlin [Fis23] discusses how to embed a covert key exchange sub protocol within a regular TLS 1.3 interaction. Even though the work does not refer to encryption, the spirit is very close to anamorphic encryption. As pointed out by the author, the embedding causes an increase in length of the ciphertext and this exposes the existence of a covert message. Horel et al. [HPRV19] also showed how to embed messages in a secret key exchange that will remain secret even to an adversary that has the secret key. Note finally that none of the early works dealt directly with the issue of how a casual sender can immediately employ anamorphism using the availability of the receiver's public key, while hiding the anamorphic message from the dictator.

2 Definitions

In this section we review the notion of a *Anamorphic Encryption* scheme [PPY22] and we propose a refined and equivalent formulation (see Definition 1 and 2) that allows to distinguish different settings depending on the nature of the double key. Then, we present the notion of public-key anamorphic encryption in Section 2.2.

An *Anamorphic Encryption scheme* is a *normal* encryption scheme $\mathsf{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ equipped with an *anamorphic* triplet $\mathsf{AME} = (\mathsf{aKG}, \mathsf{aEnc}, \mathsf{aDec})$ of algorithms. An Anamorphic Encryption scheme can be deployed in one of two modes: as a normal scheme and as an anamorphic scheme.

If Bob deploys the scheme as a normal scheme, he obtains the pair of public and secret key $(\mathtt{pk}, \mathtt{sk})$ by running the *normal* key generation algorithm KG and, as usual, pk is published. When Alice wishes to send Bob message m, she produces ciphertext ct by running the *normal* encryption algorithm Enc on input pk and m. When ct is received by Bob, it is decrypted by running the *normal* decryption algorithm Dec on input the secret decryption key sk. Thus, when deployed as

normal, an Anamorphic Encryption scheme is just a regular asymmetric encryption scheme. If the dictator comes for the secret key, Bob cannot do but surrender $\mathtt{sk}$.

If Bob deploys the scheme as anamorphic, he wants to protect the confidentiality of the communication with Alice even in the event that he is forced to surrender his secret decryption key to the dictator. In this case, Bob runs the *anamorphic* key generation algorithm aKG that returns a pair of anamorphic public-secret keys $(\mathtt{apk}, \mathtt{ask})$ along with a pair of *double keys* $(\mathtt{sdk}, \mathtt{rdk})$. Bob privately sends Alice the *sender double key* $\mathtt{sdk}$ over a private channel and keeps the *receiver double key* $\mathtt{rdk}$ private. Moreover, Bob publishes $\mathtt{apk}$ and keeps $\mathtt{ask}$ private. If requested, Bob will surrender $\mathtt{ask}$ to the dictator and pretend that it is a real secret key and that there is no receiver double key $\mathtt{rdk}$. The pair $(\mathtt{apk}, \mathtt{ask})$ is a fully functional pair of keys: if a message m is encrypted by using Enc and $\mathtt{apk}$, it can be decrypted by Dec on input $\mathtt{ask}$. Double keys are used to send messages that remain confidential even if $\mathtt{ask}$ is compromised. Specifically, whenever Alice has a message $\mathtt{amsg}$ that must remain confidential, the *anamorphic message*, she picks an innocent looking message $\mathtt{msg}$ and encrypts $(\mathtt{msg}, \mathtt{amsg})$ by running the anamorphic encryption algorithm aEnc with $\mathtt{sdk}$ and $\mathtt{apk}$. The *anamorphic ciphertext* $\mathtt{act}$ produced by aEnc has the property that it returns $\mathtt{msg}$ when decrypted with the normal decryption algorithm Dec and with key $\mathtt{ask}$; whereas it returns $\mathtt{amsg}$ when decrypted by running the anamorphic decryption algorithm aDec on input the receiver double key $\mathtt{rdk}$. In other words, the dictator will obtain $\mathtt{msg}$ and Bob will obtain $\mathtt{msg}$ and $\mathtt{amsg}$.

Clearly, the ciphertext produced by Alice must be indistinguishable from a ciphertext of $\mathtt{msg}$ produced using Enc even to an adversary that has access to $\mathtt{ask}$. The security notion formalizes this requirement. Let us now proceed more formally.

We start by defining the syntax of an *anamorphic triplet*.

Definition 1 (Anamorphic Triplet)**.** *We say that a triplet* $\mathsf{AME} = (\mathsf{aKG}, \mathsf{aEnc}, \mathsf{aDec})$ *of PPT algorithms is an* anamorphic triplet *if*

- *the* anamorphic key generation *algorithm* aKG *takes as input the security parameter* 1^λ *and returns a pair* $(\mathtt{apk}, \mathtt{ask})$ *of* anamorphic *keys and the* sender double key*,* $\mathtt{sdk}$ *and the* receiver double key $\mathtt{rdk}$*;*
- *the* anamorphic encryption *algorithm* aEnc *takes as input the anamorphic public key* $\mathtt{apk}$*, the sender double key* $\mathtt{sdk}$*, and* two *messages, the* regular message $\mathtt{msg}$ *and the* anamorphic message $\mathtt{amsg}$*, and returns an* anamorphic ciphertext $\mathtt{act}$*;*
- *the* anamorphic decryption *algorithm* aDec *takes as input the anamorphic secret key* $\mathtt{ask}$*, the receiver double key* $\mathtt{dkey}$*, and an anamorphic ciphertext* $\mathtt{act}$ *and returns a message* m*;*

and, in addition, the following correctness *requirement is satisfied*

- *for every regular message* $\mathtt{msg}$ *and anamorphic message* $\mathtt{amsg}$*, it holds that*

$$\mathsf{aDec}(\mathtt{ask}, \mathtt{rdk}, \mathtt{act}) = \mathtt{amsg}$$

  except with negligible in λ *probability, where* $((\mathtt{apk}, \mathtt{ask}), (\mathtt{sdk}, \mathtt{rdk})) \leftarrow \mathsf{aKG}(1^\lambda)$ *and* $\mathtt{act} \leftarrow \mathsf{aEnc}(\mathtt{apk}, \mathtt{sdk}, \mathtt{msg}, \mathtt{amsg})$.

Comparing with the original definition. We want to stress that the definition of [PPY22] (see also [KPP+23b]) is a special case of the definition above in which $\mathtt{sdk} = \mathtt{rdk} = \mathtt{dkey}$. In this

paper we need a fine-grained definition that distinguishes the part $\mathtt{sdk}$ of the double key that is used by the anamorphic encryption algorithm and the part $\mathtt{rdk}$ that is used by the anamorphic decryption algorithm. Specifically, we need to identify those schemes for which $\mathtt{sdk}$ is empty and thus no extra information is needed for the sender to anamorphically encrypt a message (obviously, $\mathtt{rdk}$ is non-empty). The original definition just clamped both $\mathtt{sdk}$ and $\mathtt{rdk}$ together in one double key.

2.1 Anamorphic Encryption Schemes

We are now ready to define the notion of an *Anamorphic Encryption* scheme. Roughly speaking, we will say that a secure encryption scheme $\mathsf{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is an *Anamorphic Encryption* scheme if there exists an anamorphic triplet $\mathsf{AME} = (\mathsf{aKG}, \mathsf{aEnc}, \mathsf{aDec})$ such that no PPT dictator can distinguish whether E or AME is being used, even if given access to the secret key. We formalize the notion by means of the following two games involving a dictator $\mathcal{D}$. In the real game RealG the dictator interacts with the encryption scheme deployed in normal mode whereas in the other game $\mathsf{AnamorphicG}$ the dictator interacts with the encryption scheme deployed in anamorphic mode. We require the two games to be indistinguishable; that is, we require the dictator not to be able to distinguish whether the scheme is deployed normally or anamorphically.

$\mathsf{RealG}_{\mathsf{E},\mathcal{D}}(\lambda)$

1. Set $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{KG}(1^\lambda)$
2. Return $\mathcal{D}^{\mathsf{EO}(\mathtt{pk},\cdot,\cdot)}(\mathtt{pk}, \mathtt{sk})$, where $\mathsf{EO}(\mathtt{pk}, \mathtt{msg}, \mathtt{amsg}) = \mathsf{Enc}(\mathtt{pk}, \mathtt{msg})$.

$\mathsf{AnamorphicG}_{\mathsf{AME},\mathcal{D}}(\lambda)$

1. Set $((\mathtt{apk}, \mathtt{ask}), (\mathtt{sdk}, \mathtt{rdk})) \leftarrow \mathsf{aKG}(1^\lambda)$
2. Return $\mathcal{D}^{\mathsf{Oa}(\mathtt{apk},\mathtt{sdk},\cdot,\cdot)}(\mathtt{apk}, \mathtt{ask})$, where $\mathsf{Oa}(\mathtt{pk}, \mathtt{sdk}, \mathtt{msg}, \mathtt{amsg}) = \mathsf{aEnc}(\mathtt{apk}, \mathtt{sdk}, \mathtt{msg}, \mathtt{amsg})$.

We have the following definition.

Definition 2. *We say that an encryption scheme* E *is an* Anamorphic Encryption *scheme if it is CPA secure and there exists an anamorphic triplet* AME *such that for every PPT dictator* $\mathcal{D}$ *there exists a negligible function* negl *such that*

$$\left|\Pr[\mathsf{RealG}_{\mathsf{E},\mathcal{D}}(\lambda) = 1] - \Pr[\mathsf{AnamorphicG}_{\mathsf{AME},\mathcal{D}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda).$$

Essentially, the definition says that anamorphic keys and ciphertexts are indistinguishable from regular keys and ciphertexts even to someone that has the decryption key and can ask for encryption of messages of their choice.

2.2 Public-Key Anamorphic Encryption scheme

In this section we introduce the concept of a *Public-Key Anamorphic Encryption* scheme.

Definition 3. *We say that an* Anamorphic Encryption *scheme* E *with anamorphic triplet* AME *is a* Public-Key Anamorphic Encryption *scheme if* aKG *returns an empty* $\mathtt{sdk}$.

Note that in a Public-Key Anamorphic Encryption there is no need to privately share the sender double key as it is empty. This greatly enhances the applicability of the concept of an Anamorphic Encryption encryption scheme as the sender needs not to privately communicate with the sender beforehand. Rather the sender uses the anamorphic algorithm aEnc on input the public key of the receiver and the two messages. If the public key of the receiver is indeed an anamorphic public key, the receiver will be able to obtain the anamorphic message by using the receiver double key $\mathtt{rdk}$ as input to the anamorphic decryption algorithm aDec.

For convenience we will expand the definition of a *public-key anamorphic* triplet and modify the syntax of the algorithms by removing the empty $\mathtt{sdk}$.

Definition 4 (Public-Key Anamorphic Triplet)**.** *We say that a triplet* $\mathsf{AME} = (\mathsf{aKG}, \mathsf{aEnc}, \mathsf{aDec})$ *of PPT algorithms is a* public-key anamorphic triplet *if*

- *the* anamorphic key generation *algorithm* aKG *takes as input the security parameter* 1^λ *and returns a pair* $(\mathtt{apk}, \mathtt{ask})$ *of* anamorphic *keys and a* double *key* $\mathtt{rdk}$*;*
- *the* anamorphic encryption *algorithm* aEnc *takes as input the* anamorphic public key $\mathtt{apk}$*, and* two *messages, the* regular message $\mathtt{msg}$ *and the* anamorphic message $\mathtt{amsg}$*, and returns an* anamorphic *ciphertext* $\mathtt{act}$*;*
- *the* anamorphic decryption *algorithm* aDec *takes as input the* receiver double key $\mathtt{rdk}$*, and an anamorphic ciphertext* $\mathtt{act}$ *and returns a message* m*;*

and, in addition, the following correctness *requirement is satisfied*

- *for every regular message* $\mathtt{msg}$ *and anamorphic message* $\mathtt{amsg}$*, it holds that* $\mathsf{aDec}(\mathtt{rdk}, \mathtt{act}) = \mathtt{amsg}$ *except with negligible in* λ *probability, where* $((\mathtt{apk}, \mathtt{ask}), \mathtt{rdk}) \leftarrow \mathsf{aKG}(1^\lambda)$ *and* $\mathtt{act} \leftarrow \mathsf{aEnc}(\mathtt{apk}, \mathtt{msg}, \mathtt{amsg})$.

3 Public-Key Anamorphic Scheme: The KW construction

Given the above definitions and the new model, in this section we show that the definition is not vacuous. We do this by proving that the Koppula-Waters CCA Encryption scheme [KW19] is, in fact, a *public-key* anamorphic encryption scheme. This further demonstrates that the new model is natural and is built inside a scheme which was designed without considering anamorphism.

The Koppula-Waters transformation is a general technique that takes any CPA-secure encryption scheme and turns it into a CCA-secure encryption scheme. Its security is based on standard cryptographic assumptions and on the existence of a *Hinting PRF* $\mathcal{H}$. Roughly speaking, a Hinting PRF is like a regular PRF and it takes a seed s and an argument x. The distinguishing adversary is allowed to see a sample of the output that depends on the secret seed s in a very specific sense. The adversary sees the output $y_0 = \mathcal{H}(s, 0)$ for a random seed s and argument 0. In addition, for $i = 1, \ldots, n$, the adversary receives an ordered pair $(y_{i,0}, y_{i,1})$. If $s_i = 0$, then $y_{i,0} = \mathcal{H}(s, i)$ and $y_{i,1}$ is random; if instead $s_i = 1$, the first element is random and the other is output of $\mathcal{H}$. In a Hinting PRF, the $2n + 1$ values $(y_0, (y_{1,0}, y_{1,1}), \ldots, (y_{n,0}, y_{n,1}))$ so computed are indistinguishable from $2n + 1$ random values.

A Hinting PRF is used to obtain CCA-security in the following way. A random seed s for Hint PRF $\mathcal{H}$ is chosen and the message $\mathtt{msg}$ is xored with with $\mathcal{H}(s,0)$ to give $c_0 = \mathtt{msg} \oplus \mathcal{H}(s,0)$. Then the ciphertext *signals* the seed $s = s_1 s_2 \cdots s_n$ so that the decryption algorithm can reconstruct s and peel off $\mathcal{H}(s,0)$ from c_0 to get $\mathtt{msg}$. More specifically, the public key of the KW CCA scheme contains $2n$ public keys $\mathtt{pk}_{0,1}, \mathtt{pk}_{1,1}, \ldots, \mathtt{pk}_{0,n}, \mathtt{pk}_{1,n}$ of a CPA-secure scheme and the ciphertext contains n *signals*, one for each bit of s. The i-th signal consists of $(c_{0,i}, c_{1,i}, c_{2,i})$. If $s_i = 0$ then $c_{0,i}$ is an encryption of a random v_i w.r.t. public key $\mathtt{pk}_{0,i}$ and randomness $H(s,i)$; $c_{1,i}$ is an encryption of 0^λ w.r.t. public key $\mathtt{pk}_{1,i}$ and true randomness; and $c_{2,i} = G(v_i)$ is the output of a PRG G on input v_i. If $s_i = 1$ then $c_{0,i}$ is an encryption of a random 0^λ w.r.t. public key $\mathtt{pk}_{0,i}$ and true randomness; $c_{1,i}$ is an encryption of a random v_i w.r.t. $\mathtt{pk}_{1,i}$ and randomness $H(s,i)$; and $c_{2,i} = G(v_i) \oplus t_i$. The values $t_1, \ldots, t_n$ are the tags of the ciphertexts and t_i is computed as $h_i(\mathtt{svk})$, for $i = 1, \ldots, n$, where $\mathtt{svk}$ is a signature verification key and the h_i's are pairwise independent hash functions. The ciphertext $(c_0, (c_{0,1}, c_{1,1}, c_{2,1}), \ldots, (c_{0,n}, c_{1,n}, c_{2,n}))$ is signed by using the signature key $\mathtt{ssk}$ associated with $\mathtt{svk}$.

The Koppula-Waters construction, very similarly to the Naor-Yung construction, keeps only half of the secret keys. Specifically, for each i, it keeps only the secret key $\mathtt{sk}_{0,i}$ associated with public keys $\mathtt{pk}_{0,i}$. The secret key $\mathtt{sk}_{0,i}$ is used to decrypt $c_{0,i}$ thus obtaining y_i. If $G(y_i) = c_{2,i}$, then the decryption algorithm guesses $s_i = 0$, otherwise it guesses $s_i = 1$. Then the algorithm verifies the $c_{1,i}$ for indices i for which $s_i = 1$ was guessed. That is, by using randomness $H(s,i)$, the decryption algorithm recovers $\tilde{y}_i$ and then checks that $c_{2,i} = G(\tilde{y}_i) + t_i$. Note that the encryption scheme is assumed to allow recovery of the plaintext from knowledge of the randomness used. Assuming this property is without loss of generality.

The proof consists of a sequence of hybrids that include a hybrid in which the challenge ciphertext is *ambiguous*; that is, for each i the signal $(c_{0,i}, c_{1,i}, c_{2,i})$ signals both $s_i = 0$ and $s_i = 1$. Note that information about the i-th bit of s (and hence about $H(s,0)$ the one-time pad used to mask the message $\mathtt{msg}$) is hidden in two places: which of $c_{0,i}$ and $c_{1,i}$ uses $H(s,i)$ as randomness (and this is obtained by checking which encrypts 0) and whether $c_{2,i}$ is $G(y_i)$ or $G(y_i) + t_i$. Note that the probability that a position can signal both 0 and 1 for randomly chosen h_i can be made negligible by choosing h_i from a space smaller than the space of the signature verification keys.

The proof instead uses an ambiguous ciphertext that encrypts y_i and $\tilde{y}_i$ in $c_{0,i}$ and $c_{1,i}$ and sets the tag t_i so that $t_i = G(y_i) - G(\tilde{y}_i)$. The functions h_i are chosen so that the tag $t_i = h_i(\mathtt{svk}^\star)$ that satisfy the above condition can be computed only for a specific, but randomly chosen, signature verification key $\mathtt{svk}^\star$ that will be used in the challenge ciphertext.

Let us now proceed more formally.

3.1 Hinting PRF

Let $n(\cdot)$ be a polynomial. An n-hinting PRF $\mathcal{H}$ consists of two probabilistic polynomial-time algorithms $\mathcal{H} = (\mathsf{HSetup}, \mathsf{HEval})$ with the following syntax:

- The setup algorithm HSetup takes as input the security parameter 1^λ and length parameter 1^ℓ and outputs public parameters $\mathtt{H.pp}$ and length $n = n(\lambda, \ell)$.
- The evaluation algorithm HEval takes as input public parameters $\mathtt{H.pp}$, seed $s \in \{0,1\}^n$, and index $i \in \{0, \ldots, n\}$ and outputs an ℓ-bit string.

The pseudorandomness guarantee of an hinting PRF is formalized by means of the following two experiments, $\mathsf{hprfGame}^0$ and $\mathsf{hprfGame}^1$.

$\mathsf{hprfGame}^0_{\mathcal{H},\mathcal{A}}(\lambda,\ell)$

1. $(\mathtt{H.pp}, n) \leftarrow \mathsf{HSetup}(1^\lambda, 1^\ell)$;
2. $y_0, y_{1,0}, y_{1,1}, \ldots, y_{n,0}, y_{n,1} \leftarrow \{0,1\}^\ell$;
3. return $\mathcal{A}(\mathtt{H.pp}, n, y_0, y_{1,0}, y_{1,1}, \ldots, y_{n,0}, y_{n,1})$;

$\mathsf{hprfGame}^1_{\mathcal{H},\mathcal{A}}(\lambda,\ell)$

1. $(\mathtt{H.pp}, n) \leftarrow \mathsf{HSetup}(1^\lambda, 1^\ell)$;
2. $s \leftarrow \{0,1\}^n$ and write $s = s_1 s_2 \cdots s_n$.
3. $y_0 = \mathsf{HEval}(\mathtt{H.pp}, s, 0)$;
4. for $i = 1$ to n:
   - $y_{i,s_i} = \mathsf{HEval}(\mathtt{H.pp}, s, i)$;
   - $y_{i,1-s_i} \leftarrow \{0,1\}^\ell$;
5. return $\mathcal{A}(\mathtt{H.pp}, n, y_0, y_{1,0}, y_{1,1}, \ldots, y_{n,0}, y_{n,1})$;

Definition 5. *A pair of PPT algorithms $\mathcal{H} = (\mathsf{HSetup}, \mathsf{HEval})$ is a* hinting PRF *if for every PPT adversary $\mathcal{A}$ and for every $\ell = \mathsf{poly}(\lambda)$, there exists a negligible function negl such that*

$$\left|\mathrm{Prob}\left[\mathsf{hprfGame}^0_{\mathcal{H},\mathcal{A}}(\lambda,\ell) = 1\right] - \mathrm{Prob}\left[\mathsf{hprfGame}^1_{\mathcal{H},\mathcal{A}}(\lambda,\ell) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

3.2 The encryption scheme

The KW encryption scheme uses the following cryptographic primitives and is parametrized by the parameter ℓ shared with the primitives.

1. A secure pseudorandom generator G that, on input a λ-bit seed, outputs $\ell_{\mathrm{sig}}(\lambda) + 3\lambda$ bits.
2. A CPA-secure encryption scheme $\mathsf{pE} = (\mathsf{pKG}, \mathsf{pEnc}, \mathsf{pDec})$. For security parameter λ, pE encrypts $(\lambda+1)$-bit plaintexts using $\ell(\lambda)$ bits of randomness. pE has perfect decryption correctness and randomness-decryptable ciphertexts; that is, it is possible to recover the plaintext carried by ciphertext from the public key and the randomness used to produce the ciphertext.
3. A strongly unforgeable one-time secure signature scheme $\mathcal{S} = (\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVerify})$. For security parameter λ, SKG outputs verification keys $\mathtt{svk}$ of length $\ell_{\mathrm{sig}}(\lambda)$.
4. A Hinting PRF $\mathcal{H} = (\mathsf{HSetup}, \mathsf{HEval})$. The setup algorithm HSetup on input $\mathsf{HSetup}(1^\lambda, 1^\ell)$ outputs $\mathtt{H.pp}$ and $n = \mathsf{poly}(\lambda)$. The eval algorithm HEval on input $\mathtt{H.pp}$ a seed s and an argument $0 \leq i \leq n$, outputs a string of length ℓ.

5. The pairwise independent hash function h_i is simply the linear function $a_i \cdot x + B$, for randomly chosen a_i and B.

Consider the following scheme $(\mathsf{kwKG}, \mathsf{kwEnc}, \mathsf{kwDec})$.

Key generation algorithm: $\mathsf{kwKG}(1^\lambda)$

1. Set $(\mathtt{H.pp}, n) \leftarrow \mathsf{HSetup}(1^\lambda, 1^\ell)$.
2. Run $\mathsf{pKG}(1^\lambda)$ and generate $2n$ pairs of keys $(\mathtt{ppk}_{b,i}, \mathtt{psk}_{b,i})$, for $b = 0, 1$ and $i = 1, \ldots, n$.
3. For $i = 1, \ldots, n$ randomly select $a_i \leftarrow \{0,1\}^{\ell_{\mathrm{sig}}(\lambda)+3\lambda}$.
4. Randomly select $B \leftarrow \{0,1\}^{\ell_{\mathrm{sig}}(\lambda)+3\lambda}$.
5. Set $\mathtt{cpk} = (\mathtt{H.pp}, B, (a_i, \mathtt{ppk}_{0,i}, \mathtt{ppk}_{1,i})_{i=1}^n)$ and $\mathtt{csk} = (\mathtt{psk}_{0,i})_{i=1}^n$.

Encryption algorithm: $\mathsf{kwEnc}(\mathtt{cpk}, m)$

1. Randomly select $s \leftarrow \{0,1\}^n$ and set $c = m \oplus \mathsf{HEval}(\mathtt{H.pp}, s, 0)$.
2. Randomly select $(\mathtt{ssk}, \mathtt{svk}) \leftarrow \mathsf{SKG}(1^\lambda)$.
3. For $i = 1, \ldots, n$
   (a) Randomly select $v_i \leftarrow \{0,1\}^\lambda$, and set $\tilde{r_i} = \mathsf{HEval}(\mathtt{H.pp}, s, i)$.
   (b) If $s_i = 0$, set $c_{0,i} = \mathsf{pEnc}(\mathtt{ppk}_{0,i}, 1|v_i; \tilde{r_i})$, set $c_{1,i} = \mathsf{pEnc}(\mathtt{ppk}_{1,i}, 0^{1+\lambda})$, and set $c_{2,i} = G(v_i)$.
   (c) If $s_i = 1$, set $c_{0,i} = \mathsf{pEnc}(\mathtt{ppk}_{0,i}, 0^{1+\lambda})$, set $c_{1,i} = \mathsf{pEnc}(\mathtt{ppk}_{1,i}, 1|v_i; \tilde{r_i})$, and set $c_{2,i} = G(v_i) + a_i + B \cdot \mathtt{svk}$.
4. Set $C = (c, (c_{0,i}, c_{1,i}, c_{2,i})_{i=1}^n)$, compute $\mathtt{SIG} \leftarrow \mathsf{SSign}(\mathtt{ssk}, C)$ and output ciphertext $\mathtt{cct} = (\mathtt{svk}, C, \mathtt{SIG})$.

Decryption algorithm: $\mathsf{kwDec}(\mathtt{csk}, (\mathtt{svk}, C = (c, (c_{0,i}, c_{1,i}, c_{2,i})_{i=1}^n), \mathtt{SIG})$

1. Verify $\mathtt{SIG}$ is a correct signature by running $\mathsf{SVerify}(\mathtt{svk}, C, \mathtt{SIG})$.
2. for $i = 1, \ldots, n$, compute the i-th bit d_i of d in the following way:
   Decrypt $c_{0,i}$ with $\mathtt{psk}_i$ and obtain m_i.
   If $m_i = 1|v_i$, for some $v_i \in \{0,1\}^\ell$, and $c_{2,i} = G(v_i)$, set $d_i = 0$; else set $d_i = 1$.
3. If, for $i = 1, \ldots, n$, the following checks are successful, output $m = c \oplus \mathsf{HEval}(\mathtt{H.pp}, d, 0)$; else output $m = \perp$.
   (a) set $\tilde{r_i} = \mathsf{HEval}(\mathtt{H.pp}, d, i)$ and recover m_i from $c_{d_i,i}$ using the randomness $\tilde{r_i}$;
   (b) check that $m_i \neq \perp$ and that $c_{d_i,i} = \mathsf{pEnc}(\mathtt{ppk}_{d_i,i}, m_i; \tilde{r_i})$;
   (c) finally, write $m_i = 1|v_i$ and check that
   if $d_i = 0$ then $G(v_i) = c_{2,i}$;
   if $d_i = 1$ then $c_{2,i} = G(v_i) + B \cdot \mathtt{svk} + a_i$;

3.3 KW is anamorphic

We observe that a ciphertext of the KW encryption scheme contains $2n$ ciphertexts of the underlining encryption scheme pE, $(c_{0,1}, c_{1,1}), \ldots, (c_{0,n}, c_{1,n})$. The dictator only holds secret keys for the n ciphertexts $c_{0,i}$ and this allows the dictator to compute the seed s used to compute the ciphertext and, consequently, all pseudorandom strings $\tilde{r}_i$. This implies that the dictator can decrypt all ciphertexts $c_{0,i}$, by using the receiver's secret key, and all ciphertexts that used $\tilde{v}_i$ thanks to the randomess-decryptability property. In sum, we can say that all ciphertexts $c_{1,i}$ such that $d_i = 0$ are semantically secure with respect to the dictator as they are computed with public key $\mathtt{ppk}_{1,i}$ whose corresponding secret key is not available to the dictator using randomness that is independent from the dictator's view.

Based on the observations above, we consider the following public-key anamorphic triplet $(\mathsf{akwKG}, \mathsf{akwEnc}, \mathsf{akwDec})$.

1. The anamorphic key generation akwKG executes kwKG and sets $(\mathtt{apk}, \mathtt{ask}) = (\mathtt{cpk}, \mathtt{csk})$. In addition, the secret keys $\mathtt{psk}_{1,i}$, for $i = 1, \ldots, n$, are saved as the receiver double key $\mathtt{rdk}$.

2. The anamorphic encryption algorithm akwEnc works exactly as the regular encryption scheme with the exception that the sender randomly selects i such that $s_i = 0$ and computes $c_{1,i}$ as $c_{1,i} = \mathsf{pEnc}(\mathtt{ppk}_{1,i}, 0 \circ \mathtt{amsg})$, where $\mathtt{amsg} \in \{0,1\}^\lambda$ is the anamorphic plaintext. In other words, the anamorphic message $\mathtt{amsg}$ is prefixed with a 0 and is inserted in one of the ciphertexts that are semantically secure with respect to the secret key held by the dictator.

3. The anamorphic decryption algorithm instead uses the receiver double key $\mathtt{rdk}$ that contains the secret keys $\mathtt{psk}_{1,i}$, for all i, and obtains the anamorphic plaintext $\mathtt{amsg}$ by decrypting all the the ciphertexts $c_{1,i}$ and outputting the last λ bits of the only plaintext that start with 0 and it is not equal to $0^{\lambda+1}$.

Theorem 1. *Under the assumption that KW is a CCA-secure encryption scheme, KW is a public-key anamorphic encryption scheme.*

Proof. Let us consider the anamorphic triplet $(\mathsf{akwKG}, \mathsf{akwEnc}, \mathsf{akwDec})$ above. We note that the only difference between the view of a dictator $\mathcal{D}$ in RealG and the one in $\mathsf{AnamorphicG}$ regards ciphertexts $c_{1,i}$ for i with $s_i = 0$. Indeed in RealG they are encryptions of $0^{\lambda+1}$ whereas in $\mathsf{AnamorphicG}$ one is an encryption of the anamorphic message $\mathtt{amsg}$. Suppose that $\mathcal{D}$ distinguishes the two games. Then we show how $\mathcal{D}$ can be used to break the IND-CPA security of the underlying encryption schemes pE. Consider a sequence of hybrid games H_j in which $\mathcal{D}$'s first j calls to the oracle are answered by EO and the remaining by Oa. Therefore, H_0 is $\mathsf{AnamorphicG}$ and H_p is RealG (where $p = p(\lambda)$ is a polynomial that upper bounds on the number of queries of $\mathcal{D}$ for security parameter λ). By a simple argument, there must be a $j^\star$ for which $\mathcal{D}$'s probabilities of outputting 1 differ by at least $1/\mathsf{poly}(\lambda)$.

Now consider the following IND-CPA adversary $\mathcal{A}$ that receives the challenge public key $\mathtt{ppk}$. $\mathcal{A}$ builds $\mathtt{apk}$ and $\mathtt{ask}$ for $\mathcal{D}$ by running algorithm kwKG. $\mathcal{A}$ randomly selects $i^\star$ and replaces $\mathtt{ppk}_{1,i^\star}$ with $\mathtt{ppk}$. Then $\mathcal{A}$ runs $\mathcal{D}$ on input $(\mathtt{apk}, \mathtt{ask})$ so computed and answers the first $j^\star$ queries as in EO. The $(j^\star+1)$-st query is handled in a special way. $\mathcal{A}$ receives $\mathtt{msg}$ and $\mathtt{amsg}$ from $\mathcal{D}$ and outputs as its own challenge messages $0^{\lambda+1}$ and $\mathtt{amsg}$, thus receiving the challenge ciphertext $\mathtt{pct}^\star$. Then $\mathcal{A}$ randomly selects s and if $s_{i^\star} = 1$, $\mathcal{A}$ aborts and outputs a random bit. Otherwise, the answer to $(j^\star + 1)$-st query is constructed by executing kwEnc with the value of s chosen with the only

exception that $c_{1,i^\star}$ is set equal to the challenge ciphertext $\mathtt{pct}^\star$. The remaining queries are handled by $\mathcal{A}$ by executing Oa. Finally, $\mathcal{A}$ returns $\mathcal{D}$'s output.

Observe that $\mathcal{A}$ aborts with probability exactly $1/2$ in which case its output is random. Suppose now that $\mathcal{A}$ does not abort. If $\mathtt{pct}^\star$ is an encryption of $0^{\lambda+1}$, then the answer to $(j^\star+1)$-st query is distributed as the output of EO and thus the whole view of $\mathcal{D}$ is the same as in hybrid $H_{j^\star+1}$. On the other hand, if $\mathtt{pct}$ is an encryption of $\mathtt{amsg}$ then the view of $\mathcal{D}$ is the same as in hybrid $H_{j^\star}$. Conditioned on $\mathcal{A}$ not aborting, the probability that $\mathcal{A}$ outputs 1 when $\mathtt{pct}^\star$ is an encryption of $0^{\lambda+1}$ or of $\mathtt{amsg}$ differ by $1/\mathsf{poly}(\lambda)$. By putting the two cases together, we see that $\mathcal{A}$ breaks the IND-CPA security of pE

□

4 Anamorphic Hybrid Encryption: KW as a KDEM

The public-key anamorphism of KW that we have proved in this section can be used as the basis of what we call a *hybrid anamorphic* encryption mode. This mode is very similar to hybrid encryption in the regular domain of encryption. A hybrid ciphertext is composed of two ciphertexts: an asymmetric ciphertext carrying a key K (called Key Encapsulation Method (KEM) and a symmetric ciphertext carrying the message encrypted using K (called the data encapsulation method (DEM)). In other words, in hybrid encryption, the asymmetric encryption works as a KEM and the symmetric encryption as a DEM.

Next, we claim that any public-key anamorphic encryption scheme can be used in *anamorphic hybrid encryption* as an anamorphic KEM to encapsulate the double key for the anamorphic DEM implemented by means of an anamorphic encryption scheme. Specifically, the hybrid anamorphic ciphertext consists of a public-key anamorphic ciphertext carrying the double key that is used to produce the ciphertext carrying the anamorphic message.

KW can be used in anamorphic hybrid encryption but in a much more efficient way by using a single ciphertext. Specifically, we use an approach similar to the one used in [KPP$^+$23b] for proving the anamorphism of several encryption schemes. Note that, for $i = 1, \ldots, n$, the nonce v_i is chosen at random from $\{0,1\}^\lambda$. We can then replace each v_i with a ciphertext of a symmetric encryption scheme with pseudorandom ciphertexts that carries the i-th anamorphic message encrypted with an ephemeral double key $\mathtt{sdk}$. How is the ephemeral encryption key $\mathtt{sdk}$ sent to the receiver? For this we use the public-key anamorphism we have just shown for KW.

Roughly speaking, the same KW ciphertext carries an ephemeral encryption key which is kept hidden from the dictator by using the public-key anamorphic property of KW. And then, the same ciphertext carries n anamorphic messages that are kept hidden from the dictator by using non-public-key anamorphism. This greatly improves the anamorphic efficiency (i.e., capacity) of the KW encryption scheme as summarized in the following theorem by obtaining constant rate (of anamorphic message to ciphertext size).

Let us proceed more formally and start by defining the notion of a symmetric encryption scheme $\mathsf{prE} = (\mathsf{prKG}, \mathsf{prEnc}, \mathsf{prDec})$ with *pseudorandom ciphertexts* using the following game $\mathsf{PRCtG}^\eta_{\mathsf{prE},\mathcal{A}}$, where $\eta \in \{0,1\}$, prE is a symmetric encryption scheme, and $\mathcal{A}$ is a PPT adversary . We assume that prE for security parameter λ has λ-bit keys and encrypts λ-bit plaintexts into $\ell(\lambda)$-bit ciphertexts.

$\mathsf{PRCtG}^{\eta}_{\mathsf{prE},\mathcal{A}}(\lambda)$

1. Set $K \leftarrow \mathsf{prKG}(1^\lambda)$
2. Return $\mathcal{A}^{\mathsf{OPr}^\eta(K,\cdot)}(1^\lambda)$, where

   $\mathtt{msg} \in \{0,1\}^\lambda$;

   $\mathsf{OPr}^0(K, \mathtt{msg})$ returns a randomly selected $\ell(\lambda)$-bit string;

   $\mathsf{OPr}^1(K, \mathtt{msg}) = \mathsf{prEnc}(K, \mathtt{msg})$.

Definition 6. *Let* $\mathsf{prE} = (\mathsf{prKG}, \mathsf{prEnc}, \mathsf{prDec})$ *be an IND-CPA symmetric encryption scheme. We say that* prE *has* pseudorandom ciphertexts *if for every PPT adversary* $\mathcal{A}$ *we have*

$$\left|\Pr[\mathsf{PRCtG}^0_{\mathsf{prE},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{PRCtG}^1_{\mathsf{prE},\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda).$$

Symmetric encryption schemes with pseudorandom ciphertexts are constructed assuming one-way functions. For example, consider the encryption scheme whose secret key $K \in \{0,1\}^\lambda$ is the seed of PRF $\mathcal{F} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$. To encrypt message $\mathtt{msg} \in \{0,1\}^\lambda$, one selects $r \in \{0,1\}^\lambda$ and outputs the pair $\mathtt{ct} = (r, \mathtt{msg} \oplus \mathcal{F}(K, r))$. It is easy to see that the scheme is IND-CPA secure and that the ciphertext $\mathtt{ct}$ is indistinguishable from a randomly selected string of the same length. Here we have, $\ell(\lambda) = 2\lambda$.

Next we define the following public-key anamorphic triplet $(\mathsf{kdemKG}, \mathsf{kdemEnc}, \mathsf{kdemDec})$.

1. The anamorphic key generation kdemKG coincides with akwKG. Specifically it executes kwKG and sets $(\mathtt{apk}, \mathtt{ask}) = (\mathtt{cpk}, \mathtt{csk})$. In addition, the secret keys $\mathtt{psk}_{1,i}$, for $i = 1, \ldots, n$, are saved as the receiver double key $\mathtt{rdk}$.

2. The anamorphic encryption algorithm $\mathsf{kdemEnc}$ receives in input n κ-bit anamorphic messages $\mathtt{amsg}_1, \ldots, \mathtt{amsg}_n \in \{0,1\}^\kappa$ such that $\ell(\kappa) = \lambda$. It picks the ephemeral double key $\mathtt{dkey} \leftarrow \{0,1\}^\lambda$ and computes the n anamorphic DEM ciphertexts $\mathtt{act}_i = \mathsf{prEnc}(\mathtt{dkey}, \mathtt{amsg}_i)$ and sets $v_i = \mathtt{act}_i$, for $i = 1, \ldots, n$. Moreover, the sender randomly selects i such that $s_i = 0$ and computes the anamorphic KEM ciphertext $c_{1,i}$ as $c_{1,i} = \mathsf{pEnc}(\mathtt{ppk}_{1,i}, 0 \circ \mathtt{dkey})$. The rest of the encryption proceeds as in kwEnc.

3. The anamorphic decryption algorithm instead uses the receiver double key $\mathtt{rdk}$ that contains the secret keys $\mathtt{psk}_{1,i}$, for all i, and obtains the ephemeral double key $\mathtt{dkey}$ by decrypting all the ciphertexts $c_{1,i}$ and outputting the last λ bits of the only plaintext that start with 0 and it is not equal to $0^{\lambda+1}$. Then it obtains the $v_1, \ldots, v_n$ and decrypts each of them using prDec on input $\mathtt{dkey}$ to obtain the anamorphic messages $\mathtt{amsg}_1, \ldots, \mathtt{amsg}_n$.

By using the encryption scheme with pseudorandom ciphertexts based on PRF described above, we obtain the following theorem.

Theorem 2. *Under the assumption that KW is a CCA-secure encryption scheme, the* KW *encryption scheme is a public-key anamorphic encryption scheme and can carry a message of* $\lambda n/2$ *bits in a ciphertext of size* $O(\lambda n)$ *bits.*

We note that the original KW construction only carries a λ-bit message. Also, we can save a factor of 2, by assuming that AES is a PRP and by using AES to encrypt the anamorphic messages on keys pseudorandomly derived from `dkey`. We omit further details.

We note that enhancing the efficiency of a CCA-secure scheme to carry larger messages while keeping its CCA-security is an interesting consequence of hybrid anamorphic encryption. It is a new application beyond the anamorphic adversary (i.e., the dictator). It is an interesting general question to know whether anamorphism has other new applications beyond the extra security it provides under the extreme dictatorial adversary model.

5 CCA security and anamorphic encryption

In this section we formalize the intuition as to why CCA security gives anamorphism. The aim of this section is to formalize the intuition and to show that the anamorphism of the KW construction [KW19] and of the NY paradigm [NY90] are different faces of the same phenomenon. Our goal in this section is not to provide new constructions but rather to give a deeper understanding of the ones that have been presented, and the connections between notions. Indeed, the general construction shown in this section gives a "single message anamorphic encryption" which is a weak form of anamorphic encryption that can be used to produce only one anamorphic ciphertext, which is however sufficient to demonstrate the connections at the heart of the issues we investigate here (and non-generically we get multi message schemes).

We have seen that often CCA security gives anamorphic encryption. For example, the general construction based on SPHF of [KPP$^+$23b] explains the anamorphic nature of the Cramer-Shoup encryption scheme. In this section we give an explanation for the anamorphism of CCA encryption schemes that are obtained from CPA encryption schemes and give sufficient conditions for anamorphism for the security reduction used to prove CCA security. Just as it happened for SPHF and Cramer-Shoup [KPP$^+$23b], the general construction presented in this section has some limitations (see discussion at the end of the section) that can be circumvented by the constructions based on specific CCA secure encryption schemes (like the one for the KW encryption scheme of Section 3 and the one for the Naor-Yung encryption scheme presented in [PPY22]).

On CPA-to-CCA Transformation in the Random Oracle Model. As CCA security represents the strongest and most challenging security level for PKE, achieving both this level of security and efficiency is difficult. Consequently, many practical constructions fall into the so-called random oracle model [BR94]. A quite common property in these constructions (*e.g.,* [FO99, FOPS01, Sho01, OP01, PP03] just to name a few) is that the decryption algorithm recovers both the message and the randomness. As shown in [KPP$^+$23b], this randomness recovery allows us to obtain a (symmetric) anamorphism.

In contrast, CPA-to-CCA transformations in the standard model, which we consider in this paper, do not have normally the property of randomness recovery. Consequently, it is challenging to determine whether these resulting CCA schemes are anamorphic. Our formalization of security reduction enables us to prove the anamorphism of CCA encryption in the standard model. Interestingly, the resulting schemes could potentially achieve the new (sometimes desired) form of public-key anamorphism defined here, and we explain when this happens.

5.1 The intuition

Before proceeding formally, in this section we give a rough idea of why a reduction of the CCA security of encryption scheme cE from the CPA security of pE could be useful in constructing an Anamorphic triplet for cE and in the next section we discuss why some reductions give public-key anamorphism and others do not.

Roughly speaking, the reduction plays as an adversary in the CPA security game for pE against a challenger $\mathcal{C}$ and, concurrently, as a challenger in the CCA security game for cE against an adversary $\mathcal{B}$. The aim of the reduction is to construct a successful CPA adversary for pE $\mathcal{A}$ by leveraging on the assumed CCA adversary $\mathcal{B}$ for cE, thus contradicting the CPA security of pE.

The reduction receives a public key $\mathtt{ppk}$ from the challenger $\mathcal{C}$ of the CPA game and must provide a public key $\mathtt{cpk}$ to $\mathcal{B}$ to start the CCA game. We call rKG the algorithm used to generate $\mathtt{cpk}$ from $\mathtt{ppk}$. Note that, for a successful reduction, the public key $\mathtt{cpk}$ output by rKG must be indistinguishable from a randomly selected public key. The reduction then, in its role of challenger of the CCA game, must handle two types of queries from $\mathcal{B}$: *decryption queries*, in which the reduction must decrypt a CCA ciphertext provided by $\mathcal{B}$; and *challenge queries*, in which the reduction must provide a ciphertext for one of the two messages provided by $\mathcal{B}$. On the other hand, in its role of an adversary in the CPA game, the reduction can issue challenge queries to $\mathcal{C}$, the challenger of the CPA game.

Successfully answering decryption queries means that the reduction not only produced $\mathtt{cpk}$ but also an internal state rst, (we can think of it as the random coin tosses R used by rKG), that is functionally equivalent to the secret key $\mathtt{csk}$ associated with $\mathtt{cpk}$. In addition, the reduction must be able to compute a challenge ciphertext $\mathtt{cct}$ for $\mathtt{cpk}$ after receiving a challenge ciphertext $\mathtt{pct}$ from $\mathcal{C}$. Note that here there are two messages: one, call it $\mathtt{pm}$, is contained in $\mathtt{pct}$ and the other, call it $\mathtt{cm}$, is explicitly given as input to the construction of $\mathtt{cct}$. Which of the two can play the role of the anamorphic message and which the one of the regular message? Clearly, the regular message must be the one that can be decrypted with $\mathtt{csk}$, as this is the key that the dictator will have access to. Moreover, observe that, by the CPA-security of $\mathtt{ppk}$, the information available to the reduction cannot yield $\mathtt{pm}$. Therefore it makes sense to assume that $\mathtt{pm}$ plays the role of the anamorphic message and $\mathtt{cm}$ the role of the regular message. The secret key $\mathtt{psk}$ associated with $\mathtt{ppk}$ is the double key. Note that, $\mathtt{psk}$ is not available to the reduction but that does not mean it cannot be made available to anamorphic decryption algorithm aDec. By summarizing,

- The anamorphic key generation algorithm aKG first generates a pair of keys $(\mathtt{ppk}, \mathtt{psk})$ for pE and then runs rKG on input $\mathtt{ppk}$.
- The anamorphic encryption algorithm aEnc first encrypts $\mathtt{amsg}$ using $\mathtt{ppk}$ to obtain $\mathtt{pct}$; then aEnc runs the challenge generation algorithm of the reduction, call it rChall, on input the normal message $\mathtt{msg}$ and the ciphertext $\mathtt{pct}$ to obtain the anamorphic ciphertext $\mathtt{act}$. Here we are hiding the detail that the messages encrypted is not necessarily one of those provided by $\mathcal{B}$.

There are still obstacles before we can construct an Anamorphic Encryption from a security reduction. Namely,

1. Even if rst is functionally equivalent to $\mathtt{csk}$, it might not be indistinguishable from it. This will make RealG and $\mathsf{AnamorphicG}$ distinguishable as the dictator $\mathcal{D}$ has access to the secret key of cE in both games.

2. Clearly $\mathtt{psk}$ can decrypt $\mathtt{pct}$. But this does not guarantee that ciphertext $\mathtt{pct}$ can be extracted from $\mathtt{cct}$.

We next show that these are the only obstacles to the construction of an Anamorphic triplet from a security reduction. Specifically, we will show that if the security reduction guarantees the properties in the bullets above, then it is possible to construct an Anamorphic triplet for cE. Before proceeding more formally in Section 5.3, let us dive a little bit deeper into the intuition and understand why and how the same approach gives public anamorphism (through Koppula-Waters) and non-public anamorphism (through Naor-Yung and other NIZK-based constructions).

5.2 Reductions that give Public Anamorphism

We next give an intuition to as why some reductions give public anamorphism and others do not.

A proof of CCA security shows that games ccaG^0 and ccaG^1 are indistinguishable (see next section for a formal definition of these security games). The only difference between the two games is in the challenge ciphertext that carries $\mathtt{cm}_0$ in ccaG^0 and $\mathtt{cm}_1$ in ccaG^1. At some point during the proof there will be some intermediate experiment in which the challenge ciphertext is *ambiguous*, in the sense that it could be decrypted into two different messages by two different decryption algorithms. An honestly generated public key admits ambiguous ciphertexts with only negligible probability whereas the reduction constructs a public key that is indistinguishable from honestly generated public key but it admits an ambiguous ciphertext. Clearly, the adversary should not be able to produce such a special ciphertext for otherwise it would give it as input to the decryption oracle and find out whether it is in a real game against the CCA challenger or it is playing the game against the reduction. And, indeed, the reduction possesses some special trapdoor associated with the public key that makes it possible to generate the ambiguous ciphertext.

For example, in the proof of the Naor-Yung construction (see Appendix A for a description of the Naor-Yung construction) the reduction produces a special ciphertext $(\mathtt{ct}_0, \mathtt{ct}_1, \Pi)$ in which $\mathtt{ct}_0$ and $\mathtt{ct}_1$ encrypt different messages and Π is a "proof" that they actually carry the same message. The honest decryption algorithm will decrypt $\mathtt{ct}_0$ whereas the reduction will decrypt $\mathtt{ct}_1$. If the adversary managed to produce such a ciphertext then it might distinguish the reduction from the challenger. And indeed the simulation soundness of the NIZK Π makes sure that the adversary has a negligible probability of producing such a special ciphertext. On the other hand, the reduction, unlike the adversary, possesses the trapdoor associated with the common reference string Σ used to produce the NIZK proof and can produce such a ciphertext. The Koppula-Waters construction instead adds a layer of authentication that signals the content of the message. There exists one special hidden signing key $\mathtt{ssk}$ (determined by the public key) that allows to signal both $\mathtt{cm}_0$ and $\mathtt{cm}_1$. However one must be able to sign using $\mathtt{ssk}$ in order to produce an ambiguous ciphertext.

As outlined in the previous section, the anamorphic encryption algorithm uses the algorithm rChall of the reduction that is based on the CPA-security of the underlying CPA-secure encryption scheme. In general, algorithm rChall will use the trapdoor information generated by the reduction to be able to construct the ambiguous ciphertext and the trapdoor information needed to construct the ambiguous ciphertext constitutes $\mathtt{sdk}$, the part of the double key $\mathtt{dkey}$ that is needed by the sender. This is for example the case of the Naor-Yung construction in the experiment in which rChall produces an ambiguous ciphertext $(\mathtt{ct}_0, \mathtt{ct}_1, \Pi)$ that carries two different plaintexts. Note that in game ccaG the two ciphertexts carry the same plaintext and indistinguishability is obtained by relying on CPA security. Things are different, instead, for the Koppula-Waters encryption schemes.

The reduction that is involved in the game that relies on the CPA security of the underlying scheme does not need to produce an ambiguous ciphertext and thus no trapdoor is needed. Since the trapdoor is the part $\mathtt{sdk}$ of the double key used by the sender, no trapdoor means that the sender needs no secret to encrypt the anamorphic message thus yielding a public-key anamorphic scheme.

5.3 Security Games and Reductions

In this section we review the CPA and the CCA security games, and give the formal definition of a *reduction* and of a *security reduction*. Throughout the section we fix two encryption schemes: the "CPA scheme" $\mathsf{pE} = (\mathsf{pKG}, \mathsf{pEnc}, \mathsf{pDec})$ and the "CCA scheme" $\mathsf{cE} = (\mathsf{cKG}, \mathsf{cEnc}, \mathsf{cDec})$. After defining the notion of a *reduction* and of a *security reduction* (see Definition 9 and 10) we will consider two improvements on it. The first is syntactic as it requires a special form from the output of the reduction algorithms and we introduce the notion of *Extractable Reduction* (see Definition 11 and Definition 12). Then we need to adapt the notion of reduction to work in the "hostile" environment of our anamorphic games, RealG and $\mathsf{AnamorphicG}$, where the adversary, the dictator in this case, actually obtains the secret key. This leads us to the notion of *strong security reduction* (see Definition 14).

CPA security. We start by defining the CPA security game $\mathsf{cpaG}^{\alpha}_{\mathsf{pE},\mathcal{A}}$, for $\alpha = 0, 1$, encryption scheme $\mathsf{pE} = (\mathsf{pKG}, \mathsf{pEnc}, \mathsf{pDec})$, and adversary $\mathcal{A}$ as follows.

$\mathsf{cpaG}^{\alpha}_{\mathsf{pE},\mathcal{A}}(\lambda)$

1. $(\mathtt{ppk}, \mathtt{psk}) \leftarrow \mathsf{pKG}(1^{\lambda})$;
2. Return $\mathcal{A}^{\mathsf{EO}^{\alpha}_{\mathsf{pE}}(\mathtt{ppk},\cdot,\cdot)}(\mathtt{ppk})$, where $\mathsf{EO}^{\alpha}_{\mathsf{pE}}(\mathtt{ppk}, \mathtt{pm}_0, \mathtt{pm}_1) = \mathsf{pEnc}(\mathtt{pk}, \mathtt{pm}_{\alpha})$;

We say that an adversary $\mathcal{A}$ is a *CPA-canonical* adversary if it makes exactly one query to the encryption oracle EO.

Definition 7. *We say that encryption scheme* pE *is* CPA-secure *if for every PPT CPA-canonical adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\left|\text{Prob}\left[\mathsf{cpaG}^{0}_{\mathsf{pE},\mathcal{A}}(\lambda) = 1\right] - \text{Prob}\left[\mathsf{cpaG}^{1}_{\mathsf{pE},\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

The CCA game. We next define the CCA security game $\mathsf{ccaG}^{\beta}_{\mathsf{cE},\mathcal{B}}$, for $\beta = 0, 1$, encryption scheme $\mathsf{cE} = (\mathsf{cKG}, \mathsf{cEnc}, \mathsf{cDec})$, and adversary $\mathcal{B}$.

$\mathsf{ccaG}^{\beta}_{\mathsf{cE},\mathcal{B}}(\lambda)$

1. $(\mathtt{cpk}, \mathtt{csk}) \leftarrow \mathsf{cKG}(1^{\lambda})$;
2. Return $\mathcal{B}^{\mathsf{EO}^{\beta}_{\mathsf{cE}}(\mathtt{cpk},\cdot,\cdot),\mathsf{DO}_{\mathsf{cE}}(\mathtt{csk},\cdot)}(\mathtt{cpk})$, where
   - $\mathsf{EO}^{\beta}_{\mathsf{cE}}(\mathtt{cpk}, \mathtt{cm}_0, \mathtt{cm}_1) = \mathsf{cEnc}(\mathtt{cpk}, \mathtt{cm}_{\beta})$;
   - $\mathsf{DO}_{\mathsf{cE}}(\mathtt{csk}, \mathtt{ct}) = \mathsf{cDec}(\mathtt{csk}, \mathtt{ct})$;

We say that an adversary $\mathcal{B}$ is *CCA-canonical* if it invokes the encryption oracle EO only once and it does not invoke the decryption oracle DO on the ciphertext obtained from the encryption oracle.

Definition 8. *Encryption scheme* cE *is* CCA-secure *if for every PPT canonical adversary* $\mathcal{B}$ *there exists a negligible function* negl *such that*

$$\left|\text{Prob}\left[\mathsf{ccaG}^0_{\mathsf{cE},\mathcal{B}}(\lambda)=1\right]-\text{Prob}\left[\mathsf{ccaG}^1_{\mathsf{cE},\mathcal{B}}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda).$$

Reductions. Next we define the syntax of a reduction and the notion of a security reduction. Roughly speaking, we consider a reduction Redx as sitting between a CCA adversary $\mathcal{B}$ and a CPA challenger $\mathcal{C}$. The aim of the reduction is to simulate the role of a CCA challenger for adversary $\mathcal{B}$ while playing as a CPA adversary $\mathcal{A}$ against CPA challenger $\mathcal{C}$. The combined game played by the reduction $\mathsf{RedxG}^{\alpha,\beta}$ is indexed by two parameters: α defines the game cpaG^{α} played against CPA challenger $\mathcal{C}$ and β is for game ccaG^{β} played against CCA adversary $\mathcal{B}$.

In game $\mathsf{RedxG}^{\alpha,\beta}$ reduction Redx works as follows (see also Figure 1). The reduction receives a randomly generated public key $\mathtt{ppk}$ of pE from $\mathcal{C}$. Then the reduction generates public key $\mathtt{cpk}$ of cE for $\mathcal{B}$, runs $\mathcal{B}$ on input $\mathtt{cpk}$, and replies to its queries. We denote by rKG the *reduction key generation* algorithm used to compute $\mathtt{cpk}$ along with some private information rst. Then the reduction must answer decryption queries by $\mathcal{B}$ and we denote by rAnsw the algorithm that decrypts ciphertexts provided by $\mathcal{B}$. Algorithm rAnsw takes as input the ciphertext provided by $\mathcal{B}$ as well as the private reduction info rst output by rKG. The reduction must also reply to the single encryption query $(\mathtt{cm}_0,\mathtt{cm}_1)$ issued by $\mathcal{B}$. We assume that $\mathcal{B}$ is a canonical CCA-adversary; that is, it issues exactly one encryption query and that the ciphertext produced as a reply to the encryption query is not given as input to the decryption oracle. Note that it is usual for adversaries to produce exactly one encryption query and this is the main reason why the anamorphic triplet based on Redx that we design in the next section is going to be a one-message triplet. The encryption query is served by Redx by using two algorithms. The first such algorithm $\mathsf{rSel}(\mathtt{cm}_0,\mathtt{cm}_1)$ takes as input the messages provided by $\mathcal{B}$ and returns a pair of messages $(\mathtt{pm}_0,\mathtt{pm}_1)$ that are used as arguments to the encryption oracle of the CPA game. $\mathcal{C}$ will then reply with a ciphertext $\mathtt{pct}$ of $\mathtt{pm}_\alpha$. Finally, the reply to $\mathcal{B}$'s encryption query is computed by the reduction by running algorithm rChall on input $\mathtt{cm}_\beta$, ciphertext $\mathtt{pct}$ and the reduction private information rst.

Let us now proceed more formally.

Definition 9 (Reduction)**.** *Let* cE *and* pE *be encryption schemes. A* $(\mathsf{pE},\mathsf{cE})$ reduction Redx *consists of four efficient algorithms* $\mathsf{Redx}=(\mathsf{rKG},\mathsf{rAnsw},\mathsf{rSel},\mathsf{rChall})$ *with the following syntax.*

1. *the* reduction key generation *algorithm* rKG *takes as input the public key* $\mathtt{ppk}$ *of* pE *and outputs a* reduction public key $\mathtt{rpk}$ *and a* private reduction information rst*;*

2. *the* query answering *algorithm* rAnsw *takes as input a ciphertext* $\mathtt{cct}$ *of* cE *and the private state* rst *and outputs message* $\mathtt{msg}$*;*

3. *the* message selection *algorithm* rSel *takes as input two messages* $(\mathtt{cm}_0,\mathtt{cm}_1)$ *and the reduction private information* rst *and outputs two messages* $(\mathtt{pm}_0,\mathtt{pm}_1)$*;*

4. *the* challenge *algorithm* rChall *takes as input the private state* rst*, a ciphertext* $\mathtt{pct}$ *for* pE *and a plaintext* $\mathtt{cm}$ *and constructs a ciphertext* $\mathtt{cct}$ *for* cE*.*

	$\mathcal{B}$	Redx with β	$\mathcal{C}$ with α
Key Generation			$(\mathsf{ppk}, \mathsf{psk}) \leftarrow \mathsf{pKG}(1^\lambda)$
		$(\mathsf{rpk}, \mathsf{rst}) \leftarrow \mathsf{rKG}(\mathsf{ppk})$	
	rpk		
Decryption Oracle Call	$\mathsf{DO}(\mathtt{ct}) \Rightarrow$		
		$\mathtt{msg} \leftarrow \mathsf{rAnsw}(\mathtt{ct}, \mathsf{rst})$	
	$\mathtt{msg}$		
Encryption Oracle Call	$\mathsf{EO}(\mathtt{cm}_0, \mathtt{cm}_1) \Rightarrow$		
		$(\mathtt{pm}_0, \mathtt{pm}_1) \leftarrow \mathsf{rSel}(\mathtt{cm}_0, \mathtt{cm}_1)$	
			$\mathtt{pct} \leftarrow \mathsf{pEnc}(\mathsf{ppk}, \mathtt{pm}_\alpha)$
		$\mathtt{cct} \leftarrow \mathsf{rChall}(\mathsf{rpk}, \mathtt{cm}_\beta, \mathtt{pct})$	
	$\mathtt{cct}$		

Figure 1: In game RedxG, the reduction Redx behaves as sitting between the challenger $\mathcal{C}$ and the CCA-canonical adversary $\mathcal{B}$. The reduction Redx interacts with $\mathcal{B}$ as a challenger for game ccaG^β and interacts with $\mathcal{C}$ as a CPA-canonical adversary $\mathcal{A}$ in game cpaG^α.

We next define the four *reduction* games $\mathsf{RedxG}^{\alpha,\beta}_{\mathcal{B}}$, for a CCA-canonical adversary and for $\alpha, \beta = 0, 1$. These games will be used to define the notion of a *security reduction* and to prove the properties of the anamorphic triplet that we will design. See Figure 1 for the interplay of the reduction algorithms with a CCA-canonical adversary $\mathcal{B}$ and the challenger $\mathcal{C}$ for game cpaG.

$\mathsf{RedxG}^{\alpha,\beta}_{\mathsf{cE},\mathcal{A}}(\lambda)$

1. $(\mathsf{ppk}, \mathsf{psk}) \leftarrow \mathsf{pKG}(1^\lambda)$;
2. $(\mathsf{rpk}, \mathsf{rst}) \leftarrow \mathsf{rKG}(\mathsf{ppk})$;
3. Return $\mathcal{B}^{\mathsf{rEO}^{\alpha,\beta}(\mathsf{rst},\cdot,\cdot),\mathsf{rDO}(\mathsf{rst},\cdot)}(\mathsf{rpk})$, where
   - $\mathsf{rEO}^{\alpha,\beta}(\mathsf{rst}, \mathtt{cm}_0, \mathtt{cm}_1) = \mathsf{rChall}(\mathsf{rst}, \mathtt{cm}_\beta, \mathtt{pct})$, where $\mathtt{pct} = \mathsf{pEnc}(\mathsf{pk}, \mathtt{pm}_\alpha)$ and $(\mathtt{pm}_0, \mathtt{pm}_1) \leftarrow \mathsf{rSel}(\mathtt{cm}_0, \mathtt{cm}_1, \mathsf{rst})$;
   - $\mathsf{rDO}(\mathsf{rst}, \mathtt{cct}) = \mathsf{rAnsw}(\mathtt{cct}, \mathsf{rst})$;

We have the following definition.

Definition 10 (Security Reduction.)**.** *A reduction* Redx *is a* security reduction *if for every PPT*

CCA-canonical adversary $\mathcal{B}$ there exists a negligible function negl *such that*

$$\left|\text{Prob}\left[\mathsf{RedxG}_{\mathcal{B}}^{0,\beta}(\lambda)=1\right]-\text{Prob}\left[\mathsf{ccaG}_{\mathcal{B}}^{\beta}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)$$

for $\beta=0,1$.

The definition above requires the reduction Redx to be able to simulate game $\mathsf{ccaG}_{\mathcal{B}}$ for every PPT CCA-canonical adversary $\mathcal{B}$. In a typical proof, establishing the security reduction according to Definition 10 is a crucial step and this is achieved under suitable complexity assumption.

We next prove that, if pE is CPA-Secure, then $\mathsf{RedxG}^{0,0}$ and $\mathsf{RedxG}^{1,0}$ are indistinguishable and, similarly, so are $\mathsf{RedxG}^{0,1}$ and $\mathsf{RedxG}^{1,1}$. This property is crucial for our proof of anamorphism.

Lemma 1. *Let* Redx *be a* $(\mathsf{pE},\mathsf{cE})$ *reduction. If* pE *is CPA-secure then for every PPT CCA-canonical adversary* $\mathcal{B}$ *there exists a negligible function* negl *such that*

$$\left|\text{Prob}\left[\mathsf{RedxG}_{\mathcal{B}}^{0,\beta}(\lambda)=1\right]-\text{Prob}\left[\mathsf{RedxG}_{\mathcal{B}}^{1,\beta}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)$$

for $\beta=0,1$.

Proof. For the sake of contradiction, suppose there exists PPT CCA-canonical adversary $\mathcal{B}$ that contradicts the theorem for $\beta=0$ (the case $\beta=1$ is identical). That is, $\mathcal{B}$ is such that

$$\text{Prob}\left[\mathsf{RedxG}_{\mathcal{B}}^{0,0}(\lambda)=1\right]\geq \text{Prob}\left[\mathsf{RedxG}_{\mathcal{B}}^{1,0}(\lambda)=1\right]+1/\mathsf{poly}(\lambda)$$

for some polynomial poly.

Consider the following PPT CPA-canonical adversary $\mathcal{A}$. $\mathcal{A}$ interacts with the challenger $\mathcal{C}$ of game cpaG for encryption scheme pE. $\mathcal{A}$ receives from $\mathcal{C}$ randomly generated public key $\mathtt{ppk}$, computes $(\mathtt{rpk},\mathtt{rst})\leftarrow \mathsf{rKG}(\mathtt{ppk})$, and then runs $\mathcal{B}$ on input $\mathtt{rpk}$. $\mathcal{B}$ issues two types of queries that are provided for by $\mathcal{A}$ as follows:

- decryption query for ciphertext $\mathtt{cct}$ is handled by returning $\mathsf{rAnsw}(\mathtt{rst},\mathtt{cct})$;
- encryption query for $(\mathtt{cm}_0,\mathtt{cm}_1)$ is handled by first computing $(\mathtt{pm}_0,\mathtt{pm}_1)\leftarrow \mathsf{rSel}(\mathtt{cm}_0,\mathtt{cm}_1,\mathtt{rst})$; then encryption query $(\mathtt{pm}_0,\mathtt{pm}_1)$ is issued to $\mathcal{C}$ thus receiving $\mathtt{pct}$. Then $\mathcal{A}$ sets and returns $\mathtt{cct}^{\star}\leftarrow \mathsf{rChall}(\mathtt{rpk},\mathtt{cm}_0,\mathtt{pct})$.

Finally, $\mathcal{A}$ returns $\mathcal{B}$'s output.

This ends the description of $\mathcal{A}$.

Now observe that if $\mathcal{C}$ is playing game cpaG^0, then $\mathcal{A}$ provides $\mathcal{B}$ with a view from $\mathsf{RedxG}^{0,0}$. In other words ,

$$\text{Prob}\left[\mathsf{cpaG}_{\mathcal{A}}^{0}(\lambda)=1\right]=\text{Prob}\left[\mathsf{RedxG}_{\mathcal{B}}^{0,0}(\lambda)=1\right].$$

On the other hand, if $\mathcal{C}$ is playing game cpaG^1, then $\mathcal{A}$ provides $\mathcal{B}$ with a view from $\mathsf{RedxG}^{1,0}$ and therefore

$$\text{Prob}\left[\mathsf{cpaG}_{\mathcal{A}}^{1}(\lambda)=1\right]=\text{Prob}\left[\mathsf{RedxG}_{\mathcal{B}}^{1,0}(\lambda)=1\right].$$

This contradicts the CPA security of pE. □

5.4 The reduction from the KW proof

In this section we show that the security proof of the KW encryption scheme identifies a security reduction that fits our definition. We refer to the ePrint version [KW18a] of the proof of the encryption scheme that appears in [KW19].

Specifically, we look at the proof of Lemma 4.4 on page 17. Here it is proved that hybrids H_3^β and H_4^β are indistinguishable, for $\beta = 0, 1$. In a previous sequence of lemmas, game H_3^β was proved to be indistinguishable from the CCA game ccaG^β (under appropriate security assumptions). The proof that CPA-security implies the indistinguishability of H_3^β and H_4^β contains a reduction Redx that plays ccaG^β with a CCA adversary while playing CPA game cpaG^α. The thrust of the proof is to show that if $\alpha = 0$ then the reduction is playing H_3^β which has been proved already indistinguishable from ccaG^β. In other words, the game $\mathsf{RedxG}^{0,\beta}$ is indistinguishable from ccaG^β, as required by our definition of security reduction (see Definition 10).

The reduction is as follows.

1. The rKG algorithm generates a new pair of public and secret keys for the KW encryption scheme by running the key generation algorithm kwKG with the only difference that the CPA public key $\mathtt{ppk}$ received in input is the public key $\mathtt{pk}_{1,i}$ of the KW encryption scheme for a randomly chosen i. The secret keys $\mathtt{sk}_{i,0}$ for all $\mathtt{pk}_{i,0}$ constitutes the private information rst.

2. To answer a decryption query, algorithm rAnsw simply decrypts the ciphertext by using the secret keys $\mathtt{sk}_{j,0}$ contained in the decryption key $\mathtt{csk}$ of KW as described in hybrid H_3.

3. the message selection algorithm rSel outputs a pair $(0^{\lambda+1}, 1|v)$ for a random $v \in \{0,1\}^\lambda$;

4. Finally algorithm rChall receives as input a ciphertext $\mathtt{pct}$ for one of the two messages output by rSel and constructs the ciphertext by embedding it as $c_{1,i}$.

5.5 Sufficient Conditions for Anamorphism

In this section we give sufficient conditions for a security reduction of a CCA-secure scheme $\mathsf{cE} = (\mathsf{cKG}, \mathsf{cEnc}, \mathsf{cDec})$ from a CPA-secure scheme to yield anamorphism. The two conditions are syntactic conditions in the sense that they require that extraction of cryptosystem objects from reduction objects.

The first condition, *Extended Key Generation*, requires that rKG outputs the secret key associated with the public key or, in other words, that the secret key $\mathtt{rsk}$ associated with the public key rpk output by rKG can be extracted from the random coin tosses used. Note that the internal state rst output by rKG is functionally equivalent to the secret key, in the sense that it can be used by rAnsw to answer the adversary decryption queries. The condition is thus purely syntactic.

Definition 11 (Extended Key Generation)**.** *We say that* $(\mathsf{pE}, \mathsf{cE})$ *reduction* RedxG *enjoys the* Extended Key Generation *if the output of* rKG *can be parsed as* $(\mathsf{rpk}, (\mathtt{rsk}, \mathsf{rst}))$ *and the distribution of* $(\mathsf{rpk}, \mathtt{rsk})$ *is indistinguishable from the output of* cKG*.*

See below for the consequences of this property on rAnsw.

The next condition, *Challenge Ciphertext Extractability*, requires instead that it is possible to extract the ciphertext $\mathtt{pct}$ of pE from the ciphertext produced by rChall on input $\mathtt{pct}$. We note that rChall takes, besides $\mathtt{pct}$, also the internal state rst and a message $\mathtt{msg}$.

Definition 12. *[Public-Key and Ciphertext Extractability.] We say that* $(\mathsf{pE}, \mathsf{cE})$ *reduction* RedxG *with Extended Key Generation property enjoys the* Public-Key and Ciphertext Extractability *property if there exist efficient algorithms* $\mathtt{rextCT}$ *and* $\mathtt{rextPK}$ *such that, for all messages* $\mathtt{pm}$ *and* $\mathtt{cm}$*, it holds that*

$$\begin{aligned}\text{Prob}\,[\,(\mathtt{ppk},\mathtt{psk}) \leftarrow \mathsf{pKG}(1^\lambda); (\mathtt{rpk},\mathtt{rsk},\mathsf{rst}) \leftarrow \mathsf{rKG}(\mathtt{ppk}); \mathtt{pct} \leftarrow \mathsf{pEnc}(\mathtt{ppk},\mathtt{pm});\\ \mathtt{cct} \leftarrow \mathsf{rChall}(\mathsf{rst},\mathtt{cm},\mathtt{pct}) : \mathtt{rextCT}(\mathtt{cct},\mathsf{rst}) \neq \mathtt{pct}\,] \leq \mathsf{negl}(\lambda).\end{aligned}$$

and

$$\begin{aligned}\text{Prob}\,[\,(\mathtt{ppk},\mathtt{psk}) \leftarrow \mathsf{pKG}(1^\lambda); (\mathtt{rpk},\mathtt{rsk},\mathsf{rst}) \leftarrow &\mathsf{rKG}(\mathtt{ppk});\\ &\mathtt{rextPK}(\mathtt{rpk}) \neq \mathtt{ppk}\,] \leq \mathsf{negl}(\lambda).\end{aligned}$$

We call a reduction that enjoys Extended Key Generation and Extractability an *Extractable* reduction because it allows to extract $\mathtt{rsk}, \mathtt{pct}$, and $\mathtt{ppk}$.

Finally, we introduce one more definition for a special case in which the reduction needs no private state.

Definition 13. *We say that a* $(\mathsf{pE}, \mathsf{cE})$ *an extractable reduction* $\mathsf{Redx} = (\mathsf{rKG}, \mathsf{rAnsw}, \mathsf{rChall}, \mathsf{rSel})$ *is a* public-key *reduction if* rKG *returns* $(\mathtt{rpk}, \mathtt{rsk}, \mathsf{rst})$ *with* $\mathsf{rst} = \emptyset$.

As we shall see, public-key reductions can be used to construct public-key anamorphic triplet.

Examples. We note that the two properties above are quite common and are enjoyed by the reductions used to prove security of the Naor-Yung construction [NY90], the Sahai construction [Sah99], the Lindell construction [Lin03], and the Koppula-Waters construction [KW19].

The NY construction (and its derivations) encrypts the same plaintext using two independently generated public keys $\mathtt{ppk}_1$ and $\mathtt{ppk}_2$ of a CPA-secure encryption scheme and the secret key of the scheme is $\mathtt{psk}_2$, the one associated with $\mathtt{ppk}_2$. In the reduction, $\mathtt{ppk}_1$ is set equal to the one provided as a challenge and $\mathtt{ppk}_2$ is generated along with its secret key $\mathtt{psk}_2$ by the reduction. It is thus possible for the reduction to give the adversary $\mathtt{psk}_2$ (that is the secret key of the NY encryption scheme) thus satisfying the Extended Key Generation property. The internal state of the reduction will contain the trapdoor of the NIZK used to prove well-formedness of the ciphertexts. When a challenge ciphertext $\mathtt{pct}_1$ is received by the reduction, the reduction just computes another ciphertext $\mathtt{pct}_2$ (for the public key it has generated) and it simulates a proof Π of consistency by using the trapdoor of the NIZK. The $\mathtt{cct}$ output is simply the triplet $(\mathtt{pct}_1, \mathtt{pct}_2, \Pi)$ and thus the Challenge Ciphertext Extractability also holds.

For the KW encryption scheme, we look at the reduction that is used to show to prove Lemma 4.4 (see the ePrint version of the paper [KW18b] and the brief description in the previous section). There the $\mathtt{cct}$ challenge is embedded as an inner ciphertext $c_{1,i}$, for an i such that $s_i = 1$. Extractability is thus trivial. For the Extended KG we note that the secret key for KW can be constructed as the challenge public key $\mathtt{ppk}$ is embedded as $\mathtt{pk}_{1,i}$ for some i and kwKG only retains the secret keys for $\mathtt{pk}_{0,j}$, for all j.

Strong secure reductions. We next present the *strong* versions of the games of the previous section. Specifically, in sccaG, the strong version of ccaG, and sRedxG, the strong version of the reduction game, the adversary $\mathcal{A}$ is also given the secret key $\mathtt{csk}$. We stress that since the adversary

is given the decryption key in the sccaG, it does not make sense to consider this game for formalizing a security notion regarding the plaintext encrypted by the encryption oracle EO; rather, we only use to formalize a property of the strong reduction; that is, that no PPT adversary can tell whether it is interacting with reduction or not. This notion is necessary to make the reduction work in the demanding environment of anamorphism in which the adversary actually has decryption keys. It is actually surprising that reductions used for purposes other than anamorphism enjoy this property.

Definition 14 (Strong Security Reduction.)**.** *A strong reduction* sRedx *is a* strong security reduction *if for all canonical PPT adversaries* $\mathcal{A}$ *we have that*

$$\left|\text{Prob}\left[\mathsf{sRedxG}^{0,\beta}_{\mathsf{cE},\mathcal{B}}(\lambda)=1\right]-\text{Prob}\left[\mathsf{sccaG}^{\beta}_{\mathsf{cE},\mathcal{B}}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)$$

for $\beta=0,1$*, where the games are defined as follows.*

$\mathsf{sRedxG}^{\alpha,\beta}_{\mathsf{cE},\mathcal{B}}(\lambda)$

1. $(\mathtt{ppk},\mathtt{psk})\leftarrow \mathsf{pKG}(1^\lambda)$;
2. $(\mathtt{rpk},\mathtt{rsk},\mathsf{rst})\leftarrow \mathsf{rKG}(\mathtt{ppk})$;
3. Return $\mathcal{B}^{\mathsf{rEO}^{\alpha,\beta}(\mathsf{rst},\cdot,\cdot),\mathsf{rDO}(\mathsf{rst},\cdot)}(\mathtt{rpk},\mathtt{rsk})$, where
   - $\mathsf{rEO}^{\alpha,\beta}(\mathsf{rst},\mathtt{cm}_0,\mathtt{cm}_1)=\mathsf{rChall}(\mathsf{rst},\mathtt{cm}_\beta,\mathtt{pct})$, where $\mathtt{pct}=\mathsf{pEnc}(\mathtt{pk},\mathtt{pm}_\alpha)$ and $(\mathtt{pm}_0,\mathtt{pm}_1)\leftarrow \mathsf{rSel}(\mathtt{cm}_0,\mathtt{cm}_1,\mathsf{rst})$;
   - $\mathsf{rDO}(\mathsf{rst},\mathtt{cct})=\mathsf{rAnsw}(\mathtt{cct},\mathtt{rsk})$;

$\mathsf{sccaG}^{\beta}_{\mathsf{cE},\mathcal{B}}(\lambda)$

1. $(\mathtt{cpk},\mathtt{csk})\leftarrow \mathsf{cKG}(1^\lambda)$;
2. Return $\mathcal{B}^{\mathsf{EO}^\beta(\mathtt{cpk},\cdot,\cdot),\mathsf{DO}(\mathtt{csk},\cdot)}(\mathtt{cpk},\mathtt{csk})$, where
   $\mathsf{EO}^\beta(\mathtt{cpk},\mathtt{cm}_0,\mathtt{cm}_1)=\mathsf{cEnc}(\mathtt{cpk},\mathtt{cm}_\beta)$;
   $\mathsf{DO}(\mathtt{csk},\mathtt{pct})=\mathsf{cDec}(\mathtt{csk},\mathtt{pct})$;

Note that the Extended Key Generation property guarantees that in a strong security reduction, all the ciphertexts (produced by rChall or computed using rpk) can be decrypted by using $\mathtt{rsk}$. Were this not the case, the adversary would have a very simple test to tell the reduction from the security game: just try to decrypt the challenge ciphertext obtained from the encryption oracle or try to decrypt a ciphertext produced by running the encryption algorithm on the public key received as input at the start of the game. In sccaG the decryption is always successful and so must be in sRedx. Therefore, in the description of sRedxG we let rAnsw take as input $\mathtt{rsk}$ and not rst. If $\mathsf{rst}=\emptyset$, it means that rChall does not need any trapdoor to produce $\mathtt{cct}$. As we shall see, this means that no secret information is needed by the anamorphic triplet to encrypt and thus we have public-key anamorphism.

This property is necessary to make the anamorphic triplet indistinguishable from the normal triplet. Therefore the only restriction imposed on an adversary is that the encryption oracle can be invoked only once.

Examples. Again we point out that most of the security reductions used to prove CCA security are indeed strong and satisfy Definition 14. The reduction of the Naor-Yung transform (and consequently of the derived constructions) is a strong security reduction. Roughly speaking, the difference between the reduction game and the CCA game lies in the random string used for the NIZK (that is output of the simulator in the reduction and truly random in the CCA game) and by the fact the two inner ciphertexts might carry different values. It is easy to see that the two distributions of the random string stay indistinguishable even if given the secret key of one of the CPA public keys and similarly for the two inner ciphertexts. For the KW construction indeed we observe that the difference between the two hybrids that the reduction proves indistinguishable is in the plaintext carried by $\mathtt{pct}_{1,i}$ for an i such that $s_i = 0$. The corresponding secret key $\mathtt{psk}_{1,i}$ is not part of the key output by kwKG and the ciphertext is computed using independent randomness.

Observe that the secret key of pE is not used in the strong reduction game and therefore the CPA security of pE guarantees that $\mathsf{sRedx}^{0,\beta}$ and $\mathsf{sRedx}^{1,\beta}$ are indistinguishable, for $\alpha = 0, 1$, just like proved by Lemma 1 for security reductions. Therefore,

Lemma 2. *For a strong security reduction from a CPA-secure encryption scheme* $\mathsf{pE}\ \mathsf{sRedxG}$ *we have that, for* $\beta = 0, 1$*,*

$$\left|\text{Prob}\left[\mathsf{sRedx}^{0,\beta}_{\mathsf{cE},\mathcal{B}}(\lambda) = 1\right] - \text{Prob}\left[\mathsf{sRedx}^{1,\beta}_{\mathsf{cE},\mathcal{B}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

5.6 The Anamorphic Triplet

We are now ready to describe the anamorphic triplet $(\mathsf{aKG}, \mathsf{aEnc}, \mathsf{aDec})$ for encryption scheme $\mathsf{cE} = (\mathsf{cKG}, \mathsf{cEnc}, \mathsf{cDec})$. We assume that there exist a CPA-secure encryption scheme $\mathsf{pE} = (\mathsf{pKG}, \mathsf{pEnc}, \mathsf{pDec})$ and an extractable strong $(\mathsf{pE}, \mathsf{cE})$ security reduction $\mathsf{sRedx} = (\mathsf{rKG}, \mathsf{rAnsw}, \mathsf{rSel}, \mathsf{rChall})$.

1. The anamorphic key generation algorithm $\mathsf{aKG}(1^\lambda)$ proceeds as follows.
   - Randomly select a pair consisting of public and secret key $(\mathtt{ppk}, \mathtt{psk}) \leftarrow \mathsf{pKG}(1^\lambda)$ for pE.
   - Run $\mathsf{rKG}(\mathtt{ppk})$ to obtain $(\mathtt{rpk}, (\mathtt{rst}, \mathtt{rsk}))$.
   - Return $\mathtt{apk} := \mathtt{rpk}$, $\mathtt{ask} := \mathtt{rsk}$, $\mathtt{sdk} := \mathtt{rst}$, $\mathtt{rdk} := \mathtt{rsk}$.
2. The anamorphic encryption algorithm $\mathsf{aEnc}(\mathtt{apk}, \mathtt{sdk}, \mathtt{msg}, \mathtt{amsg})$ proceeds as follows.
   - Compute $\mathtt{ppk} = \mathsf{rextPK}(\mathtt{apk})$.
   - Compute $\mathtt{pct} = \mathsf{pEnc}(\mathtt{ppk}, \mathtt{amsg})$.
   - Execute rChall on input $(\mathtt{rst}, \mathtt{msg}, \mathtt{pct})$ to produce $\mathtt{cct}$. Note that $\mathtt{rst}$ is found in $\mathtt{sdk}$.
   - Return $\mathtt{cct}$.
3. The anamorphic decryption algorithm aDec extracts $\mathtt{pct}$ from $\mathtt{cct}$ by means of the $\mathtt{rextCT}$ algorithm and decrypts it using $\mathtt{psk}$ found in $\mathtt{rdk}$.

Theorem 3 (Anamorphism from Strong Extractable Reductions)**.** *If* sRedx *is a strong extractable* $(\mathsf{pE}, \mathsf{cE})$ *reduction then* cE *is one-message Anamorphic Encryption and* $(\mathsf{aKG}, \mathsf{aEnc}, \mathsf{aDec})$ *is the anamorphic triplet for it.*

Proof. First of all, observe that aDec and aEnc always succeed by the Extractability property. Let us now prove that RealG and $\mathsf{AnamorphicG}$ are indistinguishable. Fix a dictator $\mathcal{D}$ and let $(\mathtt{msg}, \mathtt{amsg})$ be its only encryption query. Note that a dictator $\mathcal{D}$ issues no decryption query and that in this section we are interested in one-message anamorphism.

We start by observing that $\mathsf{RealG}_{\mathcal{D}}$ coincides with game $\mathsf{sccaG}^0_{\mathcal{D}}$. This can be verified by inspection. Moreover, by the definition of strong reduction, we have that $\mathsf{sccaG}^0_{\mathcal{D}}$ is indistinguishable from $\mathsf{sRedxG}^{0,0}_{\mathcal{D}}$. Therefore $\mathsf{RealG}_{\mathcal{D}}$ is indistinguishable from $\mathsf{sRedxG}^{0,0}_{\mathcal{D}}$.

Let us next consider game $\mathsf{AnamorphicG}_{\mathcal{D}}$. As a first observation, we note that this game is identical to the hybrid game $\mathcal{H}_{\mathcal{D}}$ which is obtained from game $\mathsf{sRedxG}^{1,0}_{\mathcal{D}}$ by passing $\mathtt{msg}$ to pEnc instead of $\mathtt{pm}_0$ (output of rSel). On the other hand, by CPA-security, game $\mathcal{H}_{\mathcal{D}}$ is indistinguishable from $\mathsf{sRedxG}^{1,0}_{\mathcal{D}}$. Therefore, $\mathsf{AnamorphicG}_{\mathcal{D}}$ is indistinguishable from $\mathsf{sRedxG}^{1,0}_{\mathcal{D}}$, that, in turn, is indistinguishable from $\mathsf{sRedxG}^{0,0}_{\mathcal{D}}$ thanks to Lemma 2.

□

Next, we have the following.

Theorem 4 (Public-Key Anamorphism from Strong Public-Key Reductions)**.** *If* sRedx *is a strong public-key* $(\mathsf{pE}, \mathsf{cE})$ *reduction then* cE *is one-message public-key Anamorphic Encryption and* $(\mathsf{aKG}, \mathsf{aEnc}, \mathsf{aDec})$ *is the anamorphic triplet for it.*

Proof. It is sufficient to show that the triplet is public-key and this can be easily seen from the fact that rKG returns an empty rst and thus aEnc needs no secret information. □

One-message versus many-message. As stated by Theorem 3, the construction only supports one anamorphic encryption. This is due to the technical fact that the games assume canonical adversaries that make only one query to the encryption oracle. We stress though that the construction nonetheless formally shows a surprising link between CCA security and anamorphic encryption, which is the goal of the theorem. Also, we point out that for known constructions this technical limitation to one message is either non-existent (for the KW construction) or it can be easily circumvented (for the NY paradigm). More specifically, the NY paradigm can be instantiated with NIZK proofs supporting multiple proofs on the same shared random string. And actually, most NIZKs can be made multi-proof by using the so called FLS trick [FLS90] and its upgrade for the NIZKs needed for CCA security [DDO+01].

6 Conclusions

Our work paves the way to achieving public-key anamorphism, an objective that may seem intuitively impossible, and extends the applicability of anamorphism to new useful scenarios. The new anamorphic mode then enables the KDEM encapsulation method where a ciphertext carries a KEM and DEM simultaneously; this is a novel functionality (totally beyond the original goal and reasons for anamorphism). Our novel characterization of security reductions, in turn, provides insights into the technique of achieving anamorphism via reduction: one produces non-ambiguous ciphertexts, providing public-key anamorphism, while the other produces ambiguous ciphertexts which only yields secret-key anamorphism. It seems that, overall, this structural investigation reveals interesting relationships and enriches our understanding of the design of anamorphic schemes.

We mention two possible research directions. First, it would interesting to study the possibility of having public-key anamorphic extensions in the sense of [BGH+24]. Intuitively, this seems very hard to achieve but we already have seen non-intuitive notions (such as public-key anamorphism) to be possible. We also note that our only known public-key anamorphic encryption scheme would not yield robustness [BGH+24] and it would interesting to study the feasibility of this notion.

To conclude, we note that after the initial paper [PPY22] with the Naor-Yung scheme, more recent works [KPP+23a, KPP+23b, WCHY23, CGM24, BGH+24] have confirmed the prevalence of anamorphism of various sorts in numerous cryptographic systems. We anticipate a similar trend where more research will demonstrate the prevalence of public-key anamorphism and its possible applications. Undoubtedly, this will require new insights and new possibly surprising techniques.

Acknowledgments

This work was supported in part by the France 2030 ANR Project ANR-22-PECY-003 SecureCompute.

References

[BGH+24] Fabio Banfi, Konstantin Gegier, Martin Hirt, Ueli Maurer, and Guilherme Rito. Anamorphic encryption, revisited. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part II*, volume 14652 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2024.

[BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, August 1994.

[CGM24] Dario Catalano, Emanuele Giunta, and Francesco Migliaro. Anamorphic encryption: New constructions and homomorphic realizations. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part II*, volume 14652 of *Lecture Notes in Computer Science*, pages 33–62. Springer, 2024.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222. Springer, Heidelberg, May 2004.

[DDO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001.

[Fis23] Marc Fischlin. Stealth key exchange and confined access to the record protocol data in TLS 1.3. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 2901–2914. ACM Press, November 2023.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st FOCS*, pages 308–317. IEEE Computer Society Press, October 1990.

[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, *PKC'99*, volume 1560 of *LNCS*, pages 53–68. Springer, Heidelberg, March 1999.

[FOPS01] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 260–274. Springer, Heidelberg, August 2001.

[HPRV19] Thibaut Horel, Sunoo Park, Silas Richelson, and Vinod Vaikuntanathan. How to subvert backdoored encryption: Security against adversaries that decrypt all ciphertexts. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 42:1–42:20. LIPIcs, January 2019.

[KPP+23a] Miroslaw Kutylowski, Giuseppe Persiano, Duong Hieu Phan, Moti Yung, and Marcin Zawada. Anamorphic signatures: Secrecy from a dictator who only permits authentication! In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 759–790. Springer, Heidelberg, August 2023.

[KPP+23b] Miroslaw Kutylowski, Giuseppe Persiano, Duong Hieu Phan, Moti Yung, and Marcin Zawada. The self-anti-censorship nature of encryption: On the prevalence of anamorphic cryptography. *Proc. Priv. Enhancing Technol.*, 2023(4):170–183, 2023.

[KW18a] Venkata Koppula and Brent Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. *IACR Cryptol. ePrint Arch.*, page 847, 2018.

[KW18b] Venkata Koppula and Brent Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. Cryptology ePrint Archive, Report 2018/847, 2018. `https://eprint.iacr.org/2018/847`.

[KW19] Venkata Koppula and Brent Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 671–700. Springer, Heidelberg, August 2019.

[Lin03] Yehuda Lindell. A simpler construction of cca2-secure public-key encryption under general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 241–254. Springer, Heidelberg, May 2003.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

[OP01] Tatsuaki Okamoto and David Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 159–175. Springer, Heidelberg, April 2001.

[PP03] Duong Hieu Phan and David Pointcheval. Chosen-ciphertext security without redundancy. In Chi-Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 1–18. Springer, Heidelberg, November / December 2003.

[PPY22] Giuseppe Persiano, Duong Hieu Phan, and Moti Yung. Anamorphic encryption: Private communication against a dictator. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 34–63. Springer, Heidelberg, May / June 2022.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.

[Sho01] Victor Shoup. OAEP reconsidered. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 239–259. Springer, Heidelberg, August 2001.

[WCHY23] Yi Wang, Rongmao Chen, Xinyi Huang, and Moti Yung. Sender-anamorphic encryption reformulated: Achieving robust and generic constructions. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VI*, volume 14443 of *LNCS*, pages 135–167. Springer, Heidelberg, December 2023.

A A Review of the Naor-Yung Encryption Scheme

In this appendix we describe the Naor-Yung transform [NY90] (see also [Sah99]) that, when applied to a CPA-secure public-key cryptosystem pE and a simulation sound NIZK Π for a specific polynomial-time relation $\mathsf{EqMsg}_{\mathcal{E}}$, gives a CCA public-key cryptosystem nyE.

The polynomial time relation $\mathsf{EqMsg}_{\mathsf{pE}}$ has as instances pairs of ciphertexts and public key that encrypt the same message. The witness for instance $((\mathtt{pk}_0, \mathtt{ct}_0), (\mathtt{pk}_1, \mathtt{ct}_1))$ is the triplet (r_0, r_1, m) such that

$$\mathtt{ct}_0 = \mathsf{Enc}(\mathtt{pk}_0, m; r_0) \text{ and } \mathtt{ct}_1 = \mathsf{Enc}(\mathtt{pk}_0, m; r_1).$$

1. The public key $\mathtt{nypk} = (\mathtt{pk}_0, \mathtt{pk}_1, \Sigma)$ output by key generation algorithm nyKG consists of two random and independently chosen public keys $\mathtt{pk}_0$ and $\mathtt{pk}_1$ of pE and of a random string Σ. The secret key $\mathtt{nysk} = (\mathtt{sk}_0)$ associated with $\mathtt{nypk}$ consists solely of the secret key $\mathtt{sk}_0$ associated with $\mathtt{pk}_0$.

   Note that for some cryptosystems like El Gamal it is possible to sample a public key without th secret key.

2. To encrypt message m, the encryption algorithm nyEnc first computes ciphertexts $\mathtt{ct}_0 = \mathsf{Enc}(\mathtt{pk}_0, m; r_0)$ and $\mathtt{ct}_1 = \mathsf{Enc}(\mathtt{pk}_1, m; r_1)$, using random and independent coin tosses r_0 and r_1. Then, it runs the prover's algorithm of Π to produce a proof π that $\mathtt{ct}_0$ and $\mathtt{ct}_1$ encrypt the same message. More precisely, the prover's algorithm of Π is run on input instance $((\mathtt{pk}_0, \mathtt{ct}_0), (\mathtt{pk}_1, \mathtt{ct}_1))$ and witness (r_0, r_1, m) using Σ found in $\mathtt{nypk}$ as reference string. The ciphertext $(\mathtt{ct}_0, \mathtt{ct}_1, \pi)$ is output.

3. The decryption algorithm nyDec, upon receiving ciphertext $\mathtt{ct} = (\mathtt{ct}_0, \mathtt{ct}_1, \pi)$, runs the verifier algorithm of Π to check π and, if successful, outputs m obtained by decrypting $\mathtt{ct}_0$ using $\mathtt{sk}_0$.

The anamorphic triplet for nyE $(\mathsf{aKG}, \mathsf{aEnc}, \mathsf{aDec})$ is defined as follows.

1. The anamorphic key generation algorithm aKG runs pKG and obtains $(\mathtt{pk}_0, \mathtt{sk}_0)$ and $(\mathtt{pk}_1, \mathtt{sk}_1)$. In addition, aKG runs the simulator S of the proof system Π to get string Σ and trapdoor information $\mathtt{tp}$. Then the public key is $(\mathtt{pk}_0, \mathtt{pk}_1)$ the secret key is $(\mathtt{sk}_0)$ and the double key is $\mathtt{dkey} = (\mathtt{sk}_1, \mathtt{tp})$.

2. The anamorphic encryption algorithm aEnc takes two messages $(\mathtt{msg}, \mathtt{amsg})$ and computes $\mathtt{ct}_0 = \mathsf{Enc}(\mathtt{pk}_0, \mathtt{msg})$ and $\mathtt{ct}_1 = \mathsf{Enc}(\mathtt{pk}_1, \mathtt{amsg})$ and computes proof Π by running the simulator on input $(\mathtt{pk}_0, \mathtt{ct}_0)$, $(\mathtt{pk}_1, \mathtt{ct}_1)$, random string Σ and trapdoor $\mathtt{tp}$ found in $\mathtt{dkey}$.

   We stress that $\mathtt{dkey}$ is crucial for the success of the anamorphic encryption algorithm. Indeed, the simulation soundness of the NIZK will make it possible to produce a ciphertext carrying two different messages only with negligible probability, even if Σ is output of the simulator.

3. The anamorphic decryption algorithm aDec obtains the anamorphic message from ciphertext $(\mathtt{ct}_0, \mathtt{ct}_1, \Pi)$ by decrypting $\mathtt{ct}_1$ with $\mathtt{sk}_1$ found in $\mathtt{dkey}$.