# Simple Asymmetric Anamorphic Encryption and Signature using Multi-Message Extensions

Shalini Banerjee[⋆], Tapas Pal[⋆], Andy Rupp[†⋆], Daniel Slamanig[*]

[⋆]Karlsruhe Institute of Technology, KASTEL Security Research Labs
{shalini.banerjee, tapas.pal}@kit.edu
[†]University of Luxembourg
andy.rupp@uni.lu
[*]Research Institute CODE, Universität der Bundeswehr München
daniel.slamanig@unibw.de

**Abstract**

Anamorphic encryption enables covert communication over public cryptographic channels while preserving the apparent correctness and security of the underlying cryptographic scheme, even against adversaries with access to secret keys. Classical formulations of anamorphic encryption emphasize zero-latency channels, where an anamorphic message is embedded within a single ciphertext; however, recent works show that achieving such guarantees typically requires highly structured constructions or strong, non-black-box assumptions.

In this work, we explore a complementary point in the design space by relaxing strict zero-latency and introducing $\mu$-message asymmetric anamorphic encryption (AAE), where a single anamorphic payload is distributed across a small number $\mu$ of ordinary cryptographic objects. Our first contribution is a general abstraction of $\mu$-message AAE, which captures the ability of a cryptographic primitive to jointly encode hidden information across multiple independently generated objects, while remaining indistinguishable from honestly generated ones. We formalize corresponding security notions, including indistinguishability, and resistance to key exposure, and show how these notions naturally support asymmetric anamorphic settings where the encryption-side anamorphic key may be leaked without compromising the covert channel.

Our second contribution is to demonstrate that relaxing strict zero-latency enables simple, black-box, and CCA-secure constructions of AAE from widely deployed cryptographic primitives, avoiding the need for specialized algebraic structure. In particular, we show how $\mu$-message extensions can be instantiated in several representative and practically relevant settings:

- We construct $\mu$-message AAEs from standard PKE schemes, including ElGamal and Dual-Regev, without relying on additional group- or lattice-specific sampling algorithms and achieve strong security even against adversaries with full access to secret keys.
- We show that $\mu$-message extensions compose naturally with Fujisaki–Okamoto–style transforms, yielding the first generic approach to CCA-secure AAEs from a broad class of CPA-secure schemes satisfying mild pseudorandomness properties.
- We extend the $\mu$-message framework to the identity-based encryption (IBE) and signature settings by introducing IBAE and asymmetric anamorphic signatures (AAS), illustrating that asymmetric anamorphism can be realized even in centralized, authority-driven systems and in authentication-only settings.

Finally, we introduce stream anamorphism notifiers, lightweight mechanisms that allow a receiver to efficiently locate and extract anamorphic content from a stream of cryptographic objects. Overall, our results highlight a clear trade-off between latency, deployability, and security, and show that relaxing zero-latency yields a practically relevant and theoretically clean pathway to strong anamorphic guarantees.

# Contents


**1 Introduction** **3**
- 1.1 Our Contributions . . . 4

**2 Technical Overview** **6**
- 2.1 $\mu$-Message Asymmetric Anamorphic Encryption Extensions . . . 7
- 2.2 $\mu$-Message Asymmetric Anamorphic Signature Extensions . . . 11
- 2.3 Identification of Anamorphism in Communication Streams . . . 12

**3 Preliminaries** **12**
- 3.1 Notations . . . 12
- 3.2 Computational Assumptions . . . 13
- 3.3 Pseudorandom Functions . . . 14
- 3.4 Secret Key Encryption . . . 14
- 3.5 Public Key Encryption . . . 15
- 3.6 Tag-Based Public Key Encryption . . . 16
- 3.7 Identity-Based Encryption . . . 17
- 3.8 Non-Interactive Key Exchange . . . 18
- 3.9 Signature Scheme . . . 19
- 3.10 Asymmetric Anamorphic Encryption . . . 20

**4 $\mu$-Message Asymmetric Anamorphic Encryption Extension** **23**
- 4.1 Extension of the ElGamal Encryption Scheme . . . 26
- 4.2 Extension of the Dual-Regev Encryption Scheme . . . 31
- 4.3 Extension of FO-Based PKE . . . 37

**5 Identity-Based Anamorphic Encryption Extension** **44**
- 5.1 Extension of the Boneh-Franklin BasicIdent Scheme . . . 48
- 5.2 Extension of FO-Based IBE . . . 55

**6 $\mu$-Message Asymmetric Anamorphic Signature Extension** **56**
- 6.1 Anamorphism-Friendly Signatures with Compliant Encryption . . . 58
- 6.2 Waters Signatures with Linear Encryption . . . 62
- 6.3 BBS with ElGamal and Pseudorandom Encodings . . . 64

**7 Anamorphism Notifier in Communication Streams** **66**
- 7.1 Anamorphism Notifier for Encryption . . . 66
- 7.2 Anamorphism Notifier for Signatures . . . 69

**A Informal: Robustness of Asymmetric Anamorphic Signature Extension** **77**

# 1 Introduction

Anamorphic encryption, introduced in [PPY22] and subsequently explored in [KPP+23a, BGH+24, CGM24a, PPY24a, CGM25], is a very recent concept that allows individuals to confidentially exchange messages in a very restricted setting where a powerful surveillant (typically called a "dictator") has full access to the secret keys in the system. Such a capability can foster a strong foundation for enabling privacy in the face of a powerful authoritarian government where every citizen is constantly surveilled. The basic idea behind a PKE scheme being anamorphic is that there is a second mode of encryption (the anamorphic mode) that allows encrypting an anamorphic message that can be recovered by the legitimate holder of the secret key. But from the perspective of the dictator, these ciphertexts look indistinguishable from regular ciphertexts, and so the dictator is oblivious of these anamorphic messages. Moreover, keys used for the anamorphic mode must also be indistinguishable from the ones used for the regular mode. The anamorphic functionality is achieved by having a second key (the so-called double key) that can be used by the sender and the receiver (and has to be exchanged covertly in advance). Techniques on how to secretly and efficiently exchange the double key have been recently discussed by Wan, Hang and Liu in [WHL25]. The general idea behind anamorphic encryption is to incorporate a covert channel using a symmetric or asymmetric encryption scheme on top of an existing PKE scheme. While the asymmetric setting will be the focus of this paper, we want to mention that for the symmetric setting, one usually relies either on some randomness recoverability property of the PKE [BGH+24, KPP+23b], on hybrid encryption, or an IBE-to-CCA transform [CGM24a]. Interestingly, there are well-known CCA-secure public key encryption schemes (or generic approaches to construct such schemes) such as the Naor-Yung double encryption paradigm [NY90], (RSA)-OAEP [BR95] or Cramer-Shoup [CS98] that are inherently anamorphic.

**Asymmetric anamorphism.** The most recent works in this fast-paced field focus on schemes with asymmetric anamorphic channels [CGM24a, PPY24a]. Additionally, they investigate the limitations of constructing anamorphic schemes [CGM24b, CGM25]. Arguably, asymmetric anamorphic channels, so-called asymmetric anamorphic encryption (AAE), seem to be superior to symmetric ones. In an AAE, the keys for the anamorphic mode are asymmetric, i.e., there is an encryption double key $\mathsf{dk}$ and a decryption double key $\mathsf{tk}$. This is beneficial as in the case of multiple senders, no complex key management is required in order to prevent senders from reading anamorphic messages sent to the receiver by other senders. Persiano, Phan and Yung in [PPY24a] propose a variant of the AAE notion, called public key AE (PKAE), where $\mathsf{dk} = \epsilon$, which interestingly eliminates the need of communicating the encryption double key $\mathsf{dk}$ privately to the receiver. They complement this by providing a proof of concept of the notion using the CCA-secure Koppula-Waters (KW) scheme [KW19]. Additionally, they initiate a theoretical investigation on the link between CPA-to-CCA compilers and public key anamorphism to extract a set of generic "reduction-based" properties required in the underlying PKE, satisfied by the KW scheme. Worth mentioning is also that both the PKAE constructions based on KW proposed in [PPY24a] and Naor-Yung (NY) proposed in [CGM24a] rely on a common feature of the underlying PKE scheme: the setup procedure generates multiple key pairs $(\mathsf{sk}_i, \mathsf{pk}_i)_{i\in[n]}$, while retaining only a subset of secret keys $\mathsf{sk}_j$, which makes them a natural choice for constructing PKAE by embedding anamorphic message $\mathsf{amsg}$ within ciphertext $\mathsf{ct}_j$ and allowing the receiver to retain the corresponding $\mathsf{sk}_j$ (which ideally should have been forgotten). Nevertheless, we observe that most of the well-known PKE schemes and, in particular, practically used ones do not fulfill this property.

In another recent work, Catalano, Giunta and Migliaro [CGM24a] design AAE schemes, where $\mathsf{dk} \neq \epsilon$, which they call fully-asymmetric AE schemes. Such schemes are asymmetric but require sharing the encryption key to potential senders. In particular, the authors focus on Homomorphic Anamorphic Encryption (HAE) and demonstrate that NY instantiated with any IND-CPA secure fully-homomorphic encryption scheme and fully-homomorphic NIZK [ADKL19], the Gentry-Sahai-Waters scheme [GSW13] and Cramer-Shoup Lite [CS98] are fully-asymmetric. Moreover, in a follow-up work, Catalano, Giunta and Migliaro [CGM25] show that such a type of scheme can be obtained from any IND-CPA secure PKE with sufficiently high min-entropy ciphertexts using $i\mathcal{O}$. Also, here it is worth noting that this does not cover PKE schemes that (or variants thereof) are used in practice. Importantly, all these constructions in [CGM24a] are anamorphic triplets rather than anamorphic extensions (introduced in [BGH+24]), i.e., the double keys have

to be generated at the time the normal keys are generated. However, it is desirable that schemes support *on-the-fly* anamorphic key generation, i.e., double keys can be generated based on existing keys of the normal PKE.

Moreover, interestingly, all these existing anamorphic schemes build on top of *CCA-secure* PKE schemes, but with respect to the covert channel they just achieve protection in the *CPA-sense*. This makes us think whether these limiting restrictions, i.e., reduction-based properties, being limited to anamorphic triplets rather than extensions as well as relying on CCA-secure PKEs, are inherently linked to the design of asymmetric anamorphic channels. In particular, we ask:

*Can we design simple asymmetric anamorphic encryption schemes, possibly CCA-secure ones, without such limitations and ideally based on widely deployed, practically used PKE schemes?*

Indeed, the ability to focus on simpler, *practically relevant PKE schemes (or their close variants)* increases the likelihood of acceptance by a perceptive or strategically aware dictator. We believe this is a promising direction to explore in the sense that a positive result would give us hope towards establishing AAE as a practical cryptographic primitive, while a negative result would imply relying on the existence of specialized PKE schemes, that can be outlawed by the dictator.

**Identity-based anamorphic encryption.** When considering a setting where a dictator wants to control the entire communication, it seems natural to consider identity-based encryption [BF01]. In this setting, there is an entity which generates the private keys for all the users (identities) from a compact $\mathsf{msk}$, and thus there is an inherent backdoor (this $\mathsf{msk}$) already built into the system. Thus, it seems likely that a dictator would allow such encryption mechanisms, which in addition would save the dictator from the effort of collecting all the private keys. Consequently, it seems natural to ask:

*Can we port the concept of anamorphic encryption to the identity-based encryption setting?*

**Anamorphic signatures.** Besides the dedicated study on encryption, recently Kutylowski et al. [KPP+23a] investigated whether a confidential message transmission via anamorphic channels can still work in case the dictator forbids the use of encryption, and only authentication primitives such as signatures are available [JS24, DL25a, CDHT25]. These works considered *symmetric* anamorphic channels and provided an affirmative answer to that question. After considering simpler constructions for AAE, it seems natural to ask

*Are there asymmetric anamorphic signatures (AAS), and can we design such schemes as a simple combination of existing signature and PKE schemes?*

## 1.1 Our Contributions

Classical formulations of anamorphic encryption emphasize zero-latency channels, in which an entire anamorphic message is embedded into a single ciphertext. While appealing, achieving zero-latency anamorphism in the public key setting typically requires highly structured encryption schemes, specialized randomness-recovery mechanisms, or non-black-box use of cryptographic tools. In particular, concurrent and recent works (discussed below) demonstrate that single-ciphertext anamorphism can be obtained via strong trapdoor mechanisms that enable partial or full recovery of encryption randomness—an approach that fundamentally relies on deep structural properties of the underlying scheme.

In this work, we explore a new design strategy for building simple asymmetric anamorphic encryption and signature schemes. We briefly summarize our contributions, where Table 1 provides an overview of the properties of our concrete anamorphic extensions.

- **$\mu$-message anamorphic extensions:** Instead of insisting on strict zero-latency, we introduce the new concept of $\mu$-message asymmetric anamorphic extensions (for small $\mu > 1$), where a single anamorphic payload is distributed across a small number $\mu$ of ordinary cryptographic objects (e.g., ciphertexts or signatures). We present a general abstraction of $\mu$-message AAE and formalize corresponding

security notions, including indistinguishability under key exposure and fully asymmetric settings where encryption-side anamorphic keys may be leaked without compromising the covert channel.

- **New and simple anamorphic extensions for well-known PKEs:** We construct $\mu$-message AAE schemes from simple CPA-secure PKE schemes that (or variants thereof) are used in practice. We demonstrate such extensions for the well-known ElGamal and Dual-Regev schemes with $\mu = 2$, achieving both *fully asymmetric* IND-CPA (or simply IND-CPA) and *strong* IND-CPA (sIND-CPA) security. Our Dual-Regev construction can be considered the *first fully asymmetric (s)IND-CPA secure AAE based on lattices*[1]. Note that ElGamal is used as a KEM in the popular Elliptic Curve Integrated Encryption Scheme (ECIES) [ABR99] and Dual-Regev is the basic scheme underlying many modern lattice-based schemes such as Kyber [SAB+22] (standardized as ML-KEM[2]).
  Second, many practically used schemes (such as Kyber) apply the Fujisaki-Okamoto (FO) transform [FO13] to a weakly (i.e., OW-CPA or IND-CPA) secure scheme to obtain an IND-CCA secure one. We introduce anamorphic extensions for the FO transform, thereby *achieving sIND-replayable CCA (RCCA) [CKN03] and even sIND-CCA security for the anamorphic channel*. In particular, we first show that when the ciphertext of the underlying PKE is pseudorandom in the ciphertext space (so called IND$-secure [KLR19]), then together with a suitable encoding, anamorphic messages can be embedded into the FO-transformed PKE ciphertext, achieving sIND-RCCA security. By additionally embedding a MAC value using a key generated by a non-interactive key exchange (and some additional tweaks, imperceptible by a dictator), the covert communication becomes sIND-CCA secure even w.r.t. the dictator. To the best of our knowledge, this is the first work that considers security stronger than IND-CPA for the anamorphic message (though many works in the past relied on CCA secure schemes for the normal message).
- **Introduction of identity-based anamorphic encryption (IBAE):** We initiate the study of identity-based anamorphic encryption (IBAE) and show that the Boneh-Franklin IBE (BF-IBE) [BF01] admits a 2-message anamorphic extension, satisfying fully asymmetric IND-CPA and strong IND-CPA security. Moreover, we show that the FO-approach for constructing an anamorphic channel can be equivalently applied to the IBE setting, covering CCA secure *standardized* [ETS22] variants of the BF and the Sakai-Kasahara IBEs.
- **Introduction of asymmetric anamorphic signatures (AAS):** We introduce the concept of *public key* anamorphic signatures and present first instantiations by combining well-known signature and PKE schemes in a way that embedding anamorphic messages is simple. In particular, we *generalize* our approach for constructing such anamorphic extensions by defining the properties of *anamorphism-friendly signatures* and *compliant encryption schemes* and providing a corresponding *master theorem*. We showcase our framework for Waters signatures and linear encryption as well as Boneh-Boyen-Sacham (BBS) signatures, being in the process of *standardization*, and FO-ElGamal encryption thereby achieving security beyond IND-CPA for the latter.
- **Stream anamorphism notifier:** Since $\mu$-message extensions lead to anamorphic messages being distributed over multiple cryptographic objects (ciphertexts or signatures), we investigate how to efficiently signal to a dedicated receiver where in a stream of cryptographic objects anamorphic material is embedded. To this end, we formally introduce the concepts of a stream ciphertext anamorphism notifier (SCAN) and a stream signature anamorphism notifier (SSAN) and show how to efficiently realize this concept.

**Concurrent and independent works.** Concurrent and recent lines of research have illustrated a tension in balancing latency, deployability on standard primitives, and strong security guarantees. Deo and Libert [DL25b] established that the Dual-Regev encryption scheme supports public key anamorphic encryption, which relies on a non-black-box encoding of the anamorphic message into the encryption randomness, followed by a direct security proof tailored to the specific structure of the scheme. Very recently, Do et al. [DPPY26] explores PKAEs, including Dual-Regev, for polynomially sized message space under the as-

[1] Concurrent and subsequent works [DL25b, DPPY26] builds PKAE for Dual-Regev, which are discussed below.
[2] https://doi.org/10.6028/NIST.FIPS.203

| **AAE** | | | | | |
|---|---|---|---|---|---|
| Our Work | PKE | $\mu = \#\mathsf{ct}$ | $\widehat{\mathcal{M}}$ | | Security |
| Sec. 4.1 | ElGamal | 2 | restricted | | IND-CPA+sIND-CPA |
| Sec. 4.2 | DualRegev | 2 | restricted | | IND-CPA+sIND-CPA |
| Sec. 4.3 | FO-HashedElG | 4 | unrestricted | | sIND-RCCA |
| Sec. 4.3 | FO-HashedElG | 7 | unrestricted | | sIND-CCA |
| **IBAE** | | | | | |
| Our Work | IBE | $\mu = \#\mathsf{ct}$ | $\widehat{\mathcal{M}}$ | | Security |
| Sec. 5.1 | BF | 2 | restricted | | IND-CPA+sIND-CPA |
| Sec. 5.2 | FO-BF | 4 | unrestricted | | sIND-RCCA |
| **AAS** | | | | | |
| Our Work | PKE | Sig | $\mu = \#\sigma$ | $\widehat{\mathcal{M}}$ | Security |
| Sec. 6.2 | LinEnc | Waters | 3 | unrestricted | IND-CPA |
| Sec. 6.3 | ElGamal | BBS | 4 | unrestricted | IND-CPA |
| Sec. 6.3 | FO-ElG | BBS | 5 | unrestricted | sIND-RCCA+IND-CCA |

Table 1: Our anamorphic encryption and signature schemes. Here, we denote by $\#\mathsf{ct}$ and $\#\sigma$, the total number of (regular) ciphertexts or signatures of the underlying PKE, IBE or Sig schemes needed for transmitting an anamorphic message; by $\widehat{\mathcal{M}}$, the anamorphic message space where "restricted" means polynomial-sized and "unrestricted" means exponential-sized; by "Security" the security provided by the anamorphic channel.

sumption that the underlying encryption scheme is (strongly) trapdoor randomness-extractable. Under this assumption, the authors obtain zero-latency PKAEs via rejection-sampling techniques applied to high-min-entropy randomness. As discussed in [CGM25], this approach crucially relies on sufficient min-entropy of the sampled randomness and may fail for specially designed encryption schemes whose ciphertexts exhibit low min-entropy for certain messages.

More recently, several works have investigated fundamental limitations of anamorphic encryption. In particular, [CGM24a, CGM25, DG25, ABG+25] study black-box constructions of anamorphic encryption and identify inherent upper bounds on message length, as well as classes of encryption schemes that cannot support anamorphism at all. These results investigated that achieving zero-latency public key anamorphism in full generality is impossible without additional structure or assumptions.

At the same time, achieving stronger security notions such as CCA security is typically accompanied by increased ciphertext size or additional structural constraints, even in conventional encryption. In the anamorphic setting, this tension is further amplified by the requirement that the ciphertext distribution remain indistinguishable even under key exposure. As a consequence, when restricted to simple and tightly constrained primitives—such as those likely to be mandated or approved by a dictator—simultaneously achieving zero-latency and strong security guarantees becomes particularly challenging, due to the limited capacity of such schemes to accommodate additional hidden structure.

Overall, recent works pursue zero-latency public key anamorphism via powerful randomness-recovery trapdoors and scheme-internal structural assumptions. Our work takes a different and orthogonal approach: by allowing a minimal and explicit relaxation in latency, we obtain simple constructions, broad applicability to standardized schemes, and strong CCA-level security guarantees that are difficult to achieve in the single-ciphertext setting. Taken together, our results highlight a principled trade-off between latency, deployability, and security, and show that $\mu$-message anamorphism provides a theoretically clean and practically compelling pathway to strong anamorphic guarantees. Designing CCA-secure AAE schemes that simultaneously achieve zero-latency (or even finding an optimal $\mu$) remains an intriguing open problem.

## 2 Technical Overview

We present the main ideas and selected technical details of our constructions.

## 2.1 $\mu$-Message Asymmetric Anamorphic Encryption Extensions

We have already discussed that constructing AAE (or PKAE) is known to work when making an additional assumption such as when relying on PKE where the key generation algorithm samples various key pairs of the underlying CPA-secure PKE schemes and a *subset* of the secret keys is sufficient for decryption, which gives rise to a public key anamorphic channel. Nevertheless, only theoretically interesting schemes are known to provide such a feature. Moreover, the schemes in [CGM24a] appear inherently limited to anamorphic triplets rather than extensions, despite being far from practical.

Alternatively, we could take inspiration from recent works [KPP+23a,KPP+23b] constructing secret key anamorphic encryption schemes (i.e., $\mathsf{dk} = \mathsf{tk}$) using the randomness-recovering properties of PKE schemes. We recall that for such a scheme the secret key can recover the random coins sampled during encryption. Let $r$ be the randomness used to encrypt $\mathsf{msg}$ by $\mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}; r) = \mathsf{ct}$ then there is a randomness recovery algorithm that extracts $r$ along with $\mathsf{msg}$ using $\mathsf{sk}$. To create an anamorphic channel using this property, one needs a symmetric key encryption scheme $\mathsf{SKE}$ to encrypt an anamorphic message $\mathsf{amsg}$ and use $\mathsf{SKE.Enc}(\widetilde{\mathsf{sk}}, \mathsf{amsg})$ as the random coins $r$ while computing $\mathsf{ct}$. The secret key $\widetilde{\mathsf{sk}}$ plays the role of the double key. If the SKE scheme produces pseudorandom ciphertexts then the dictator remains oblivious about the presence of such an anamorphic message inside $\mathsf{ct}$ having access to $(\mathsf{pk}, \mathsf{sk}, \mathsf{ct})$. A naive way to make this work in the case of AAE is to replace the underlying SKE having pseudorandom ciphertexts with a PKE having the same property.

However, our approach towards AAE is not to use any specific property of the PKE but to devise an approach that is capable of covering various well-known PKE schemes that (or variations thereof) are used in practice. Our design goal is to rely on a *single* PKE scheme both for the regular and anamorphic messages. It is however not clear a priori how to embed the anamorphic message $\mathsf{amsg}$ into a single ciphertext $\mathsf{ct}$ of the underlying PKE scheme such that decryption recovers both $\mathsf{msg}$ and $\mathsf{amsg}$ without relying on aforementioned assumptions or properties. To get around this apparent constraint, we embed an anamorphic message $\mathsf{amsg}$ within *multiple* ciphertexts of the underlying PKE. We formalize this as $\mu$-message asymmetric anamorphic extension and consequently a ciphertext $c$ containing an anamorphic message is spread over $\mu$ (not necessarily consecutively sent) ciphertexts $\mathbf{act} = (\mathsf{act}_1, \ldots, \mathsf{act}_\mu)$ of the underlying PKE. This can be seen as distributing a ciphertext of the PKE carrying $\mathsf{amsg}$ over $\mu$ ciphertexts containing regular messages $(m_1, \ldots, m_\mu)$ generated with the *same* PKE. To formally capture this, we additionally introduce the notion of AAE-compatibility, which requires to generate double-keys *on the fly*, and importantly *ciphertext extractibility*, demanding that the existence of an efficient extractor function $F$ such that the anamorphic message $\mathsf{amsg}$ can be computed by the decryption algorithm of the PKE on ciphertext $F(\mathbf{act})$, i.e., $\mathsf{PKE.Dec}(\mathsf{tk}, F(\mathbf{act})) = \mathsf{amsg}$. This property provides several advantages: $(i)$ it enables seamless deployment on top of existing PKE implementations without modifying decryption software or hardware, $(ii)$ it preserves backward compatibility with systems already standardized around the PKE, and $(iii)$ it increases stealthiness, as the decryption algorithm is indistinguishable from that of the underlying scheme, making the existence of the anamorphic channel harder to detect.

A few comments are in order: Our AAE formalization can be seen as a generalization of existing AAE (where $\mu = 1$) and we note that for an $n$-bit anamorphic message $\mathsf{amsg}$ a trivial solution would be to embed it into $\mu = n$ regular ciphertexts. However, this becomes non-trivial if we consider $1 \leq \mu < n$ and in particular we want to make $\mu$ as small as possible. As seen in Table 1, we actually achieve very small values for $\mu$.

We note that Wang, Chen, Huang and Yung [WCHY23] proposed the notion of $\ell$-receiver anamorphic encryption where an $\ell$-bit $\mathsf{amsg}$ is embedded using a *symmetric* anamorphic channel across $\ell$ regular PKE ciphertexts such that the $i$th ciphertext encodes the $i$th bit of $\mathsf{amsg}$ by relying on pseudorandom PKEs. On the contrary, we encode a *single* $\mathsf{amsg}$ using an *asymmetric* anamorphic channel across $\mu$ (we achieve this with $\mu = 2$) regular ciphertexts using the *same* PKE for both the regular and anamorphic messages without relying on any specific property of PKE or additional assumptions such as pseudorandom PKE ciphertexts. Next, we demonstrate a 2-message AAE extension for the ElGamal and the Dual-Regev encryption schemes.

**AAE based on ElGamal.** We consider the ElGamal scheme with public key $\mathsf{pk} = g^{\mathsf{sk}}$, ciphertext $C = (g^t, g^{t \cdot \mathsf{sk} + \mathsf{msg}})$ encrypting a message $\mathsf{msg}$ using a random integer $t$. One could wonder whether it is possible

to embed an ElGamal ciphertext $(g^{\kappa}, g^{\kappa\cdot\mathsf{tk}+\mathsf{amsg}})$ into a *single* ElGamal ciphertext $(r_1 = g^t, c_1 = g^{t\cdot\mathsf{sk}+\mathsf{msg}})$. Since $t$ is uniformly random over $\mathbb{Z}_q$, it is not possible to extract something meaningful if we try to embed $g^{\kappa\cdot\mathsf{tk}+\mathsf{amsg}}$ into $r_1$. On the other hand, both $g^{\kappa}$ and $r_1$ are statistically close. Hence, we can directly set $t = \kappa$ for recovering the correct anamorphic message during decryption. Consequently, the only possible option left is to somehow encode $g^{\kappa\cdot\mathsf{tk}+\mathsf{amsg}}$ into $c_1$ with $t = \kappa$. Thus, we have the following equation $\kappa \cdot \mathsf{tk} + \mathsf{amsg} = \kappa \cdot \mathsf{sk} + \mathsf{msg}$. Given a pair of messages $\mathsf{msg}, \mathsf{amsg}$, this equality cannot hold with high probability when $\mathsf{tk}, \mathsf{sk}$ are independently and uniformly sampled from $\mathbb{Z}_q$. As a result, we try to embed the ciphertext $(g^{\kappa}, g^{\kappa\cdot\mathsf{tk}+\mathsf{amsg}})$ into $\mu = 2$ ElGamal ciphertexts.

More specifically, we can set the double keys as $\mathsf{dk} \coloneqq (\mathsf{dk}_1 = g^{\alpha}, \mathsf{dk}_2 = g^{\alpha\cdot\mathsf{sk}})$ and $\mathsf{tk} \coloneqq \alpha$. Embedding an anamorphic message $\mathsf{amsg}$ into two ciphertexts for messages $\mathsf{msg}_1$ and $\mathsf{msg}_2$ will result in $\mathsf{act}_1 = (r_1, c_1) = (g^{\kappa_1}, \mathsf{pk}^{\kappa_1} \cdot \mathsf{msg}_1)$ and $\mathsf{act}_2 = (r_2, c_2) = (\mathsf{dk}_1^{\kappa_1} \cdot g^{\mathsf{amsg}}, \mathsf{dk}_2^{\kappa_1} \cdot \mathsf{pk}^{\mathsf{amsg}} \cdot \mathsf{msg}_2)$. Decrypting the normal messages works as usual and the anamorphic message can be recovered by calling ElGamal decryption on $(r_1, r_2)$ and finding $\log g^{\mathsf{amsg}}$. We prove fully-asymmetric IND-CPA security, basically requiring IND-CPA security for both the regular and the covert channel (cf. Definition 21), and sIND-CPA security (cf. Definition 21), requiring IND-CPA security for the covert channel even for adversaries given $\mathsf{sk}$ as input, of the AAE. To show fully-asymmetric IND-CPA security, we first rely on the DDH assumption with the instance $(g^{\kappa_1}, g^{\mathsf{tk}}, g^{\kappa_1\cdot\mathsf{tk}})$ to replace $r_2$ with a random element $r_2 = g^{\kappa_2}$ for some uniformly random integer $\kappa_2$, hiding the anamorphic message $\mathsf{amsg}$ within the ciphertext (in the dictator's view, guaranteeing anamorphic security). At this stage, both $\mathsf{act}_1$ and $\mathsf{act}_2$ become two independent ElGamal ciphertexts encrypting $\mathsf{msg}_1$ using $\kappa_1$ and $\mathsf{msg}_2$ using $\kappa_2$ respectively. Therefore, the ElGamal encryption scheme under the public key $\mathsf{pk}$ hides $\mathsf{msg}_1, \mathsf{msg}_2$, ensuring fully asymmetric IND-CPA security of the AAE. Further, we show sIND-CPA security assuming the DDH assumption where the adversary gets $\mathsf{sk}$ along with $\mathsf{pk}$ and $\mathsf{dk}$. See 4.1 for details.

**AAE based on Dual-Regev.** Before sketching the idea behind our AAE construction based on the Dual-Regev cryptosystem, we first recall the basic scheme: the key generation algorithm samples a vector $\mathbf{k} \leftarrow \{0,1\}^m$, a matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, and returns $\mathsf{pk} \coloneqq (\mathbf{A}, \mathbf{b} = \mathbf{A}\cdot\mathbf{k})$ and $\mathsf{sk} \coloneqq \mathbf{k}$. To encrypt a message $\mathsf{msg}$, the encryption algorithm chooses a vector $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, a vector $\mathbf{e} \leftarrow \chi^m$, and an integer $y \leftarrow \chi$, where $\chi$ is an LWE noise distribution, and computes ciphertext as $(\mathbf{r}^T, c) \coloneqq (\mathbf{s}^T\cdot\mathbf{A} + \mathbf{e}^T, \mathbf{s}^T\cdot\mathbf{b} + y + \mathsf{msg}\cdot\lfloor\frac{q}{2}\rfloor \mod q)$. Decryption is simply checking whether $|c - \mathbf{r}^T\cdot\mathbf{k}| \le \frac{q}{4}$. To design an AAE associated with the Dual-Regev construction, we again replace the random choices $(\mathbf{s}_1, \mathbf{s}_2)$ made during encryption of two regular messages $\mathsf{msg}_1$ and $\mathsf{msg}_2$ with specially chosen values. Intuitively, one might expect to find a solution by taking an approach identical to our ElGamal-based AAE. Nevertheless, such a straightforward adaptation does not work, which we now account in more detail. Recall that in our ElGamal-based AAE, we use $\mu = 2$ regular ciphertexts to embed anamorphism by setting the randomness $\kappa_2 = \alpha\kappa_1 + \mathsf{amsg}$, where $\kappa_1$ is uniformly random in $\mathbb{Z}_q$, such that $\mathsf{amsg}$ could be recovered by treating $(r_1 = g^{\kappa_1}, r_2 = g^{\kappa_2})$ as an ElGamal ciphertext and executing regular ElGamal decryption. This worked fine as an ElGamal ciphertext indeed contains two group elements. However, if we proceed in a similar manner with our Dual-Regev anamorphic construction, i.e., bias the randomness $\mathbf{s}_2$ in $\mathbf{r}_2^T = \mathbf{s}_2^T\cdot\mathbf{A} + \mathbf{e}_2^T$ (first component of second regular Dual-Regev ciphertext) such that $\mathsf{amsg}$ is encoded within, then we are not able to recover $\mathsf{amsg}$ since there is no way to extract $\mathbf{s}_2$ from $\mathbf{r}_2$. One could aim for a better approach by encoding $\mathsf{amsg}$ in $\mathbf{s}_2^T\cdot\mathbf{A} + \mathbf{e}_2^T$ and treating $(\mathbf{r}_1^T, \mathbf{r}_2^T)$ as a regular Dual-Regev ciphertext. But this again fails as a Dual-Regev ciphertext comprises of a vector $\mathbf{r}$ and a scalar $c$.

We need a slightly different strategy to embed the anamorphic message into $\mathbf{r}_2$. Let $\widehat{\mathbf{b}}$ be the encryption double key $\mathsf{dk}$ such that $\widehat{\mathbf{b}} = \mathbf{A}\widehat{\mathbf{k}}$ where $\widehat{\mathbf{k}} \leftarrow \{0,1\}^m$ is the decryption double key $\mathsf{tk}$. As in the case of ElGamal-based AAE, the first ciphertext component encrypting $\mathsf{amsg}$ is the first ciphertext component of $\mathsf{act}_1 = (\mathbf{r}_1^T, c_1) \coloneqq (\mathbf{s}_1^T\cdot\mathbf{A} + \mathbf{e}_1^T, \mathbf{s}_1^T\cdot\mathbf{b} + y_1 + \mathsf{msg}_1\cdot\lfloor\frac{q}{2}\rfloor \mod q)$. To recover the second ciphertext component $\mathbf{s}_1^T\cdot\widehat{\mathbf{b}} + \widehat{y} + \mathsf{amsg}\cdot\lfloor\frac{q}{2}\rfloor$ encrypting $\mathsf{amsg}$ from $\mathsf{act}_2 = (\mathbf{r}_2^T, c_2) \coloneqq (\mathbf{s}_2^T\cdot\mathbf{A} + \mathbf{e}_2^T, \mathbf{s}_2^T\cdot\mathbf{b} + y_2 + \mathsf{msg}_2\cdot\lfloor\frac{q}{2}\rfloor \mod q)$, our idea is to sample $\mathbf{s}_2$ such that the first entry of $\mathbf{r}_2^T$ is $\mathbf{s}_1^T\cdot\widehat{\mathbf{b}} + \widehat{y} + \mathsf{amsg}\cdot\lfloor\frac{q}{2}\rfloor$. More precisely, $\mathbf{s}_2$ is sampled as follows:

$x_i \leftarrow \mathbb{Z}_q$ for $i \in [1, n-1]$ , $\mathbf{s}_2^T = [x_1, \ldots, x_n]$ where

$$x_n = \left( -\sum_{i=1}^{n-1} x_i a_i + \mathbf{s}_1^T \cdot \widehat{\mathbf{b}} + \widehat{y} + \mathsf{amsg} \cdot \lfloor \frac{q}{2} \rfloor - e_{2,1} \right) \cdot a_n^{-1} \mod q$$

where $\mathbf{a} = (a_1, \ldots, a_n)$ is the first column of $\mathbf{A}$ and $e_{2,1}$ is the first entry of $\mathbf{e}_2$. Therefore, the first entry of $\mathbf{r}_2^T$ becomes $\widehat{c} = \mathbf{s}_1^T \cdot \widehat{\mathbf{b}} + \widehat{y} + \mathsf{amsg} \cdot \lfloor \frac{q}{2} \rfloor \mod q$. Thus, one can efficiently extract the regular Dual-Regev ciphertext $(\mathbf{r}_1^T, \widehat{c})$ and recover $\mathsf{amsg}$ using $\mathsf{tk} = \widehat{\mathbf{k}}$. We prove fully asymmetric IND-CPA and sIND-CPA security of the AAE. We first show that under the $\mathsf{DLWE}$ assumption $\mathbf{s}_1^T \cdot \widehat{\mathbf{b}} + \widehat{y}$ is uniformly distributed over $\mathbb{Z}_q$, ensuring uniformity of $x_n$ over $\mathbb{Z}_q$ in the adversary's view (even in the dictator's view as it does not have $\mathsf{tk} = \widehat{\mathbf{k}}$, guaranteeing anamorphic security). As a result, $\mathsf{act}_1, \mathsf{act}_2$ become independent regular Dual-Regev-ciphertexts encrypting $\mathsf{msg}_1, \mathsf{msg}_2$ respectively. In case of fully asymmetric IND-CPA security, the updated $\mathsf{act}_1, \mathsf{act}_2$ hide $\mathsf{msg}_1, \mathsf{msg}_2$ under the $\mathsf{DLWE}$ assumption. See 4.2 for details.

**Anamorphic clans and stronger security.** While the entire revolutionary theme of anamorphic cryptography is built on top of a totalitarian government regime, it is quite natural to imagine a setting where secret rebellious communities exist within. A prominent use-case would be an underground community liaising secretly to advocate freedom of citizens from the dictatorship rule. Additionally, the members of such communities must be subject to background checks to avoid infiltration by the government, and hence managed by a leader; we call such communities *anamorphic clans*.
Having introduced the concept of anamorphic clans, consider a scenario where the encryption double key $\mathsf{dk}$ of an anamorphic clan member is revealed to the dictator. A dictator could simply procure $\mathsf{dk}$ when getting distributed within the clan, or by corrupting its member(s). Intuitively, one might expect the dictator to forcefully demand the corresponding decryption double key $\mathsf{tk}$. However, the dictator may have a stronger motive of silently surveilling the communication of the alleged clan member, and hence not demand $\mathsf{tk}$ immediately. It could also be the case that the dictator is unable to link the procured $\mathsf{dk}$ to a clan member and hence wishes to investigate who is covertly communicating with whom. This motivates us to study *stronger* security notions beyond the usual IND-CPA notion, where the adversary knows both $\mathsf{dk}$ and the secret key of the regular encryption scheme $\mathsf{sk}$ (later also called $\mathsf{ask}$). For passive security, we call this notion sIND-CPA security. Additionally, we consider strong active-security notions such as sIND-CCA, where the adversary may query $\mathsf{aDec}(\mathsf{tk}, \mathbf{act})$ on any $\mathbf{act}$ except the challenge ciphertext, and sIND-RCCA, where queries decrypting to a challenge message are rejected.

**FO-based AAE with active security.** Another goal of this paper is to build AAE for popular IND-CCA secure schemes using the same idea of a multi-message extension. We study the use of the popular FO transformation [FO13] which is particularly relevant in the construction of post-quantum PKEs such as Kyber. We recall that FO encrypts a random message $r$ with a weakly secure (e.g., IND-CPA secure) PKE using randomness derived from a hash function $H$ on $H(r, \mathsf{msg})$ with $\mathsf{msg}$ being the message that one wants to encrypt. The random message $r$ is then used as a basis to derive a key $G(r)$, with $G$ being another hash function, for a symmetric encryption (SKE) scheme which then encrypts $\mathsf{msg}$. The overall scheme then achieves IND-CCA security in the random oracle model. The basic idea of our construction is to use two FO-layers, an inner and outer layer of FO encryption. Basically, the inner-layer is used to encrypt the anamorphic message $\mathsf{amsg}$ where the PKE scheme and the SKE scheme are required to have pseudorandom ciphertexts. Then we use a pseudorandom encoding/decoding (e.g,. Elligator Squared [Tib14] in case of elliptic curves) to encode the inner-layer ciphertext into the message space of the PKE of the outer-layer FO encryption. Due to the property of the FO-transformation, i.e., randomness recoverability, we can thus use the randomness obtained from decrypting the outer-layer to decode this into a ciphertext of the inner-layer and decrypt. Interestingly, this simple construction already achieves sIND-RCCA security: essentially, this is because of the non-malleability of the inner-layer challenge ciphertext $c^*$ (containing the challenge message). Mauling the outer-layer ciphertext $\mathbf{act}^*$ (e.g., by reordering components, re-encrypting using $\mathsf{sk}$ and $\mathsf{pk}$, etc.) will either result in $c' \neq c^*$ and thus in a decryption failure or in $c' = c^*$ which will be rejected by the sIND-RCCA $\mathsf{aDec}$-oracle. Hence, this oracle does not provide any information about the challenge

message. sIND-CCA security is not achieved by this construction, as using $\mathsf{sk}$ the adversary can extract $c^*$ from $\mathbf{act}^*$ and re-embed it into a new outer-layer ciphertext $\mathbf{act} \neq \mathbf{act}^*$ which will be decrypted by the sIND-CCA $\mathsf{aDec}$-oracle. To fix this, we extend our construction by computing a MAC over the outer-layer $\mathbf{act}^*$ and bind it to the inner-layer $c^*$ by means of tag-based (FO) encryption. This ensures that $c^*$ can only be embedded in the original $\mathbf{act}^*$. Of course, all this needs to be done without modifying the outer-layer FO encryption, i.e., the MAC and its key also need to be embedded obliviously into the FO ciphertexts. See Section 4.3 for details.

**Identity-based anamorphic encryption (IBAE).** The identity-based encryption (IBE) paradigm appears to be natural choice for regimes where centralized control and surveillance are paramount. In an IBE system, the dictator could generate the master key pair $(\mathsf{mpk}, \mathsf{msk})$ and derive secret user keys $\mathsf{sk}_{\mathsf{id}}$ for any public identity at any point in time, enabling efficient and silent surveillance. Alternatively, the dictator may compel an IBE key generation authority to surrender its master secret key. Crucially, this setting eliminates the need of the dictator to store users' secret keys, which simplifies key management and enhances scalability compared to traditional public key settings.

Hence, exploring the potential of anamorphic channels in IBE is very attractive. Instead of targeting simple asymmetric anamorphic channels within an IBE, we consider anamorphic IBE channels within the regular IBE, i.e., we covertly embed IBE ciphertexts within IBE ciphertexts. This is motivated the notion of anamorphic clans mentioned before, where the anamorphic clan lead will decide who can join the clan and communicate covertly with the other members. To this end, it can generate a double master key-pair $(\mathsf{dmpk}, \mathsf{dmsk})$ for the IBE and derive and distribute the double user secret keys $\mathsf{dsk}_{\mathsf{id}}$ to the clan members. In this way, a new clan member needs only to meet the clan lead to obtain $\mathsf{dsk}_{\mathsf{id}}$, rather than exchange double public keys with all members as in the AAE setting.

We formalize the above intuition by introducing the notion of IBAE-compatibility which captures two properties. The first *on-the-fly double key generation* property captures the essence that a double master key pair $(\mathsf{dmpk}, \mathsf{dmsk})$ can be generated on-the-fly using only the knowledge of $\mathsf{mpk}$, and can be deployed in two different dictatorial settings : (1) *authority-before-dictatorship* where the dictator coerces an existing IBE key generation authority to obtain the IBE master keys $(\mathsf{mpk}, \mathsf{msk})$, and later this authority can freshly generate double master key pair $(\mathsf{dmpk}, \mathsf{dmsk})$ by independently executing the IBE setup algorithm and using its knowledge of $\mathsf{mpk}$, eventually acting as the anamorphic clan leader, and (2) *authority-after-dictatorship* where the dictator itself generates the IBE master keys $(\mathsf{mpk}, \mathsf{msk})$, and an anamorphic clan leader can independently sample double master key pair $(\mathsf{dmpk}, \mathsf{dmsk})$ using only $\mathsf{mpk}$ (which is made public by the dictator). The second *ciphertext extractibility* property allows the anamorphic IBE decryption algorithm to be essentially the same as the regular IBE decryption algorithm, which additionally offers seamless deployment on existing systems, and enhances stealthiness since the anamorphic decryption algorithm remains indistinguishable from the underlying scheme.

We introduce definitions for secure anamorphic IBE extensions and present a concrete instantiation for the seminal Boneh-Franklin (BF) scheme [BF01]. Let us briefly recall the BF $\mathsf{BasicIdent}$ scheme: the master public key is $\mathsf{mpk} = (\mathbb{G}, \mathbb{G}_T, P, e, q, P_{\mathsf{pub}} = P^{\mathsf{msk}}, \mathsf{H}_1, \mathsf{H}_2)$, the private key for $\mathsf{id}$ is $\mathsf{sk}_{\mathsf{id}} = \mathsf{H}_1(\mathsf{id})^{\mathsf{msk}}$, a ciphertext for $\mathsf{id}$ is $(r, c) = (P^{\kappa}, \mathsf{msg} \oplus \mathsf{H}_2(g_{\mathsf{id}}^{\kappa}))$, where $\kappa \leftarrow \mathbb{Z}_q^*, g_{\mathsf{id}} = e(\mathsf{H}_1(\mathsf{id}), P_{\mathsf{pub}})$. The decryption algorithm simply computes $c \oplus \mathsf{H}_2(e(\mathsf{sk}_{\mathsf{id}}, r))$ and extracts $\mathsf{msg}$.

A 2-IBAE for the BF-IBE, works roughly as follows. Unlike the anamorphic extension of the ElGamal encryption scheme, where for two ciphertexts the first components $(r_1, r_2)$ represent an ElGamal encryption of the anamorphic message, embedding $\mathsf{amsg}$ within two regular BF ciphertexts is not that straightforward. In a BF ciphertext $\mathsf{act}_2 = (r_2 = P^{\kappa_2}, c_2 = \mathsf{msg}_2 \oplus \mathsf{H}_2(g_{\mathsf{id}}^{\kappa_2}))$, the group element $g_{\mathsf{id}}^{\kappa_2}$ is not explicitly available, where $g_{\mathsf{id}} = e(\mathsf{H}_1(\mathsf{id}), P^{\mathsf{amsk}})$. Rather, it only appears as an input to the hash function $\mathsf{H}_2$. Therefore, if we set $\kappa_2$ following the ElGamal encryption extension, then it seems impossible to recover the anamorphic message $\mathsf{amsg}$ due to the one-wayness of $\mathsf{H}_2$. At this point, the only solution appears to be encoding $\mathsf{amsg}$ outside the hash function $\mathsf{H}_2$. Consequently, we use rejection sampling such that $\mathsf{amsg}$ could be extracted from $c_2$. More specifically, we sample $\kappa_2$ subject to $\mathsf{H}_2(g_{\mathsf{id}}^{\kappa_2}) = \mathsf{amsg} \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}}^{\kappa_1})$ where $\widehat{g}_{\mathsf{id}} = e(\mathsf{H}_1(\mathsf{id}), P^{\mathsf{pdmsk}})$. We then treat $(r_1 = P^{\kappa_1}, \widehat{c} \coloneqq \mathsf{amsg} \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}}^{\kappa_1}))$ as a BF ciphertext and apply the regular decryption algorithm of BF

using $\mathsf{dsk}_{\mathsf{id}} = \mathsf{H}_1(\mathsf{id})^{\mathsf{dmsk}}$ to recover $\mathsf{amsg}$. Using this technique, we can convey anamorphic messages of at most $\mathrm{O}(\log \lambda)$ bits while keeping encryption polynomial-time.

Let $\mathsf{dmsk} = \alpha$ be the double master secret key. To show the anamorphic security, we rely on the DBDH assumption with the instance $(P^r, P^\alpha, P^{\kappa_1}, e(P,P)^t)$ where $t$ can be either $r\alpha\kappa_1$ or random. Once $Q_{\mathsf{id}}$ is set as $\mathsf{H}_1(\mathsf{id}) = P^r$ (by viewing $\mathsf{H}_1$ as a random oracle), we can write $\widehat{g}_{\mathsf{id}}^{\kappa_1} = e(P^{r\cdot\mathsf{dmsk}}, P^{\kappa_1}) = e(P,P)^t$ which is uniformly distributed over $\mathbb{G}_T$ under the DBDH assumption. Since $\mathsf{msk}$ is known we can still simulate $g_{\mathsf{id}}^{\kappa_1} = e(P^{r\cdot\mathsf{msk}}, P^{\kappa_1})$ for $\mathsf{act}_1$. Finally, as $\mathsf{H}_2$ is viewed as a random oracle, $\mathsf{amsg} \oplus \mathsf{H}_2(P^t)$ can be replaced by a random bit, ensuring that $\mathsf{amsg}$ is hidden from the dictator's view. We present the details of fully asymmetric IND-CPA and strong IND-CPA security proofs for the IBAE (which are proven depending on the IND-CPA security of the basic BF-IBE scheme) in Section 5.1.

Similar to AAE, we then explore the use of the FO-transformation to create an anamorphic IBE extension which achieves strong sIND-RCCA security and allows for $\lambda$-bit anamorphic messages. Standardized Boneh-Franklin and Sakai-Kasahara are covered by our results. See 5.2 for details.

## 2.2 $\mu$-Message Asymmetric Anamorphic Signature Extensions

Anamorphic signatures are a recent concept [KPP+23a] but, so far, they are not considered in the context of asymmetric anamorphic channels. Our aim is to investigate such asymmetric anamorphic signatures (AAS) based on simple combinations of well-known signature and PKE schemes.

Similar to our AAE approach, we define $\mu$-message anamorphic extension of a signature scheme, i.e., we embed an anamorphic message into multiple signatures. Besides anamorphic security, i.e., the distributions of regular and anamorphic signatures must be indistinguishable, we also define an IND-CPA notion. In contrast to encryption, however, now we only have to care about $\mathsf{amsg}$ (and not also the regular message) being hidden from someone knowing $\mathsf{dk}$. We also consider a IND-CCA variant of this notion as well as sIND-CCA notion where the adversary additionally receives the signing key $\mathsf{ask}$.

Towards such constructions, our main observation is that there are well-known schemes where signatures contain some uniformly random components and the remaining components can be computed from the former in a "black-box way" (e.g., without knowing their discrete logarithms). This opens up the possibility to embed pseudorandom ciphertexts (from an IND$-CPA secure PKE) into these random signature components. The number of random signature components and the number of ciphertext components, then controls how many signatures are required to transmit one anamorphic message. We put this intuition into a generic formal framework of anamorphism-friendly signatures and compliant encryption which we rigorously formalize. Depending on the security of the compliant PKE, we achieve either IND-CPA or sIND-RCCA, and if the signature scheme satisfies strong unforgeability (i.e., SUF-CMA) also (non-strong) IND-CCA security for anamorphic messages. See 6.1 for details.

We then show how to instantiate this framework based on Water's signatures [Wat05] together with linear encryption [BBS04] (Section 6.2) as well as BBS signatures [BBS04, TZ23], currently being standardized, together with (FO-)ElGamal encryption (Section 6.3), latter with the help of the Elligator Squared encoding [Tib14].

To provide an intuition, for Water's signatures the hash is computed as $H(\mathsf{msg}) = h_0 \prod_{i=1}^{n} h_i^{m_i}$, where the $m_i$ is the $i$'th bit of message $\mathsf{msg}$. A signature is $(g^r, g^{\alpha\beta} H(\mathsf{msg})^r)$ for random $r$, secret key $g^{\alpha\beta}$ and public key $(g^\alpha, g^\beta, h_0, \ldots, h_n)$. Also recall that in linear encryption the public key is $\mathsf{pk} = (u, v, h) \in \mathbb{G}^3$ and the secret key $(x, y) \in \mathbb{Z}_q$ such that $u^x = v^y = h$ holds. Encrypting a message $\mathsf{msg} \in \mathbb{G}$ amounts to choosing random $\kappa_1, \kappa_2$ and computing ciphertext $(c_1, c_2, c_3) = (u^{\kappa_1}, v^{\kappa_2}, h^{\kappa_1+\kappa_2} \cdot \mathsf{msg})$ which looks pseudorandom under the decision linear (DLIN) assumption and decryption computes $\mathsf{msg} = c_3 \cdot (c_1^x \cdot c_2^y)^{-1}$. When making the assumption that every user generates its own parameters for the Water's hash and thus knows $a_i$ such that $h_i = g^{a_i}$ for $i \in \{0, \ldots, n\}$ and this is tolerated by the dictator[3], we can construct a simple AAS. Using the knowledge of these discrete logarithms allows us to transmit an anamorphic message $\mathsf{amsg}$ by embedding one ciphertext component of a ciphertext $(c_1, c_2, c_3)$ encrypting $\mathsf{amsg}$ into the first components of three Water's signatures.

[3]For the second construction based on BBS we do not require such an assumption

### 2.3 Identification of Anamorphism in Communication Streams

We consider the problem of the designated receiver of an anamorphic message to efficiently identify anamorphic ciphertexts within a sequence of standard ciphertexts transmitted over a communication channel, which is particularly important when using $\mu$-message anamorphic extensions as introduced in this paper. Our setting assumes the use of a connection-oriented transport layer protocol, which guarantees reliable and in-order delivery of messages. Typical examples include the transmission control protocol (TCP) [Pos81,Bra89], which underlies the majority of Internet traffic; QUIC [IT21], a modern transport protocol designed for low-latency, connection-oriented communication over UDP; and higher-layer record protocols such as the TLS record layer [DR08,Res18], which build upon transport guarantees to provide cryptographic confidentiality and integrity. In all these cases, *assuming a passive adversary/dictator*, ciphertexts will arrive at the receiver in precisely the order in which they were produced by the sender.

In this setting, we introduce stream ciphertext anamorphism notifiers (SCAN), a mechanism that can be used by the sender and designated receiver to mark and identify (obliviously to a dictator) anamorphic ciphertexts within such ordered communication streams. As we can assume that ciphertexts carrying part of $\mathsf{amsg}$ arrive in order (and only a small constant number, *known* by the receiver, of ciphertexts are involved), it suffices to covertly mark the start of this sequence, provided that those ciphertexts are sent back-to-back by the sender. To this end, a pseudorandom function (PRF) can be leveraged as follows: Considering a stream of ciphertexts $(\mathsf{ct}_1, \mathsf{ct}_2, \ldots)$, the sender (equipped with the shared PRF key $K$ which can be part of the receiver's $\mathsf{dk}$) computes $r_k \leftarrow \mathsf{PRF}_K(\mathsf{ct}_{k-1})$, and uses it as the randomness for the encryption of the next message $\mathsf{msg}_k$ to mark the start of a sequence $(\mathsf{ct}_{k+1}, \ldots, \mathsf{ct}_{k+\mu})$ of anamorphic objects at position $k+1$. The receiver can scan for this covert signal by constantly checking whether $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}, \mathsf{msg}_i; \mathsf{PRF}(K, \mathsf{ct}_{i-1}))$.

This mechanism enables a faster recovery of the anamorphic messages as it does *not* require to trial-decrypt any potential anamorphic ciphertext (i.e., any $\mu$-sequence of consecutive $\mathsf{ct}_i$ ciphertexts) in the stream. This is different from the robustness notion introduced in [BGH+24], which ensures that it is hard for an adversary to find any message whose normal encryption would, when passed to the anamorphic decryption algorithm, produce a non-$\perp$ result. Thus, a robust scheme may guarantee that every normal ciphertext unmistakably indicates "no covert content" upon anamorphic decryption, but without a notifier the receiver would still need to attempt decryption on every ciphertext in a large stream to find the covert one; conversely, a notifier could mark the position of covert content without guaranteeing that malformed or normal ciphertexts will be cleanly distinguished by anamorphic decryption.

Importantly, our signalling technique is generic and applies to any anamorphic encryption scheme, including our $\mu$-message AAE and IBAE schemes; see Section 7. Similarly, Section 7.2 defines stream signature anamorphism notifiers (SSAN), based on pseudorandom functions, for identifying anamorphic signatures in streams for our $\mu$-message AAS schemes.

## 3 Preliminaries

### 3.1 Notations

Let $\lambda \in \mathbb{N}$ denote the security parameter. We use PPT to denote a randomized algorithm running in time $a \cdot \lambda^c$ for some constants $a$ and $c$. A function $f : \mathbb{N} \to \mathbb{R}^+$ is called negligible in $\lambda$, if it grows slower than $\lambda^{-c}$ for every constant $c$. We use $\mathsf{negl}(\lambda)$ to denote a generic negligible function. We denote the set of positive integers upto $n$ as $[n] \coloneqq \{1, \ldots, n\}$. For a finite set $S$, we use $x \leftarrow S$ to denote the uniformly sampling of an element $x$ from $S$. For a probabilistic algorithm $\mathcal{A}$, we let $y \leftarrow \mathcal{A}(x)$ denote the process of running $\mathcal{A}(x)$ with fresh randomness and assigning the result to $y$. We denote $\mathcal{D}_1 \approx_c \mathcal{D}_2$ to represent computational indistinguishability between two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$. We represent empty strings using $\epsilon$. We use lowercase bold letters to denote vectors (e.g. $\mathbf{k}$) and uppercase bold letters for matrices (e.g. $\mathbf{A}$). We strictly follow the convention of all vectors $\mathbf{k}$ being column vectors, and denote the corresponding row vector as $\mathbf{k}^T$.

## 3.2 Computational Assumptions

Next, we recall the various hardness assumptions used in our constructions. We begin with the DDH and the DBDH assumptions, which are used in our ElGamal and Boneh-Franklin-based anamorphic encryption schemes.

**Definition 1 (The DDH Assumption [Bon98])** *Let $\lambda \in \mathbb{N}$ be the security parameter and let $q$ be a polynomial in $\lambda$. Let $\mathbb{G} = \langle g \rangle$ be a group of order $q > 2^\lambda$. Then for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$, the following holds:*

$$|\Pr[a, b \leftarrow \mathbb{Z}_q : \mathcal{A}(g^a, g^b, g^{ab}) = 1] - \Pr[a, b, c \leftarrow \mathbb{Z}_q : \mathcal{A}(g^a, g^b, g^c) = 1]| \leq \mathsf{negl}(\lambda)$$

*where the probability is taken over the coin tosses of $\mathcal{A}$ and random choices of $a, b, c$ and $g$.*

**Definition 2 (The DBDH Assumption [BF01])** *Let $q \in \mathbb{N}$ be a large prime and let $(\mathbb{G}, \mathbb{G}_T, g, e, q)$ be a pairing group, where $g$ is a generator of $\mathbb{G}$. Then for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$, the following holds:*

$$\begin{aligned}|\Pr[a, b, c \leftarrow \mathbb{Z}_q : \mathcal{A}(g^a, g^b, g^c, e(g, g)^{abc}) = 1] - \\ \Pr[a, b, c, t \leftarrow \mathbb{Z}_q : \mathcal{A}(g^a, g^b, g^c, e(g, g)^t) = 1]| \leq \mathsf{negl}(\lambda)\end{aligned}$$

*where the probability is taken over the coin tosses of $\mathcal{A}$, and random choices of $a, b, c, t, g$.*

Next, we recall the Leftover Hash Lemma [ILL89] and the DLWE assumption [Reg09] used in our Dual-Regev based AAE constructions.

**Lemma 1 (Leftover Hash Lemma [ILL89])** *Let $\mathcal{H}$ be a universal hash function family consisting of function $h : \mathcal{X} \to \mathcal{Y}$. Let $X, Z$ be random variables such that $\mathbb{H}_\infty(X|Z) \geq \log |\mathcal{Y}| + 2\log(1/\epsilon)$ for some $\epsilon > 0$ and $X$ is supported over $\mathcal{X}$. Let $H, Y$ be uniformly random and independent over $\mathcal{H}, \mathcal{Y}$ respectively. Then $\boldsymbol{SD}((H, H(X), Z), (H, Y, Z)) \leq \epsilon$.*

**Definition 3 (The DLWE Assumption [Reg09])** *Let $q \in \mathbb{N}$ be a large prime and let $n, m \in \mathbb{N}$. Let $\chi$ be a noise distribution over $\mathbb{Z}_q$. The $(n, q, \chi)$- DLWE assumption of dimension $m$ states that the following distributions are computationally indistinguishable:*

$$(\mathbf{A}, \mathbf{s}^T \cdot \mathbf{A} + \mathbf{e}^T) \overset{c}{\approx} (\mathbf{A}, \boldsymbol{u}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi^m, \mathbf{u} \leftarrow \mathbb{Z}_q^m$$

We now state the CDH problem used in our AAS construction based on Water's signatures.

**Definition 4 (The CDH Problem [DH21])** *Let $q \in \mathbb{N}$ be a large prime and let $\mathbb{G} = \langle g \rangle$ be a multiplicative cyclic group of order $q$. Let $a, b \leftarrow \mathbb{Z}_q$. Then the CDH problem is to compute $g^{ab}$.*

We now recall the DLIN assumption by Boneh, Boyen and Shacham [BBS04] used in our Waters signature based asymmetric anamorphic signature.

**Definition 5 (The DLIN Assumption [BBS04])** *Let $q \in \mathbb{N}$ be a large prime and let $\mathbb{G}$ be a multiplicative cyclic group of order $q$, where $u, v, h$ are uniformly random generators of $\mathbb{G}$. Let $a, b$ be uniformly random elements of $\mathbb{Z}_q$. Then for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$, the following holds:*

$$\begin{aligned}|\Pr[u, v, h \leftarrow \mathbb{G} : \mathcal{A}(u, v, h, u^a, v^b, h^{a+b}) = 1] \\ - \Pr[u, v, h, \eta \leftarrow \mathbb{G} : \mathcal{A}(u, v, h, u^a, v^b, \eta) = 1]| \leq \mathsf{negl}(\lambda)\end{aligned}$$

*where the probability is taken over the uniformly random choice of parameters to $\mathcal{A}$, and over the coin tosses of $\mathcal{A}$.*

## 3.3 Pseudorandom Functions

**Definition 6 (Pseudorandom Function Family)** *A collection of functions*

$$\mathcal{F} = \{F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}\}$$

*with key space* $\mathcal{K} = \{0,1\}^\lambda$*, input space* $\mathcal{X} = \{0,1\}^n$ *and output space* $\mathcal{Y} = \{0,1\}^m$ *is called a* pseudorandom function family *(PRF) if:*

1. *There exists a polynomial-time algorithm that on input* $(k, x)$ *outputs* $F(k, x)$*.*
2. *For every probabilistic polynomial-time distinguisher* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\left| \Pr_{k \leftarrow \{0,1\}^\lambda} \left[\mathcal{A}^{F(k,\cdot)}(1^\lambda) = 1\right] \; - \; \Pr_{F \leftarrow \mathcal{R}} \left[\mathcal{A}^{F(\cdot)}(1^\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda),$$

*where* $\mathcal{R}$ *denotes the set of all functions from* $\{0,1\}^n$ *to* $\{0,1\}^m$*.*

## 3.4 Secret Key Encryption

**Definition 7 (Secret Key Encryption)** *We say that a triplet* $\Pi_{\mathsf{SKE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *of PPT algorithms with a message space* $\mathcal{M}$ *is an IND-CPA secure secret key encryption (SKE) scheme if the following holds:*

$\underline{\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda) :}$ *The key generation algorithm takes as input the security parameter* $1^\lambda$*, and outputs a secret key* $\mathsf{sk}$*.*

$\underline{\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathsf{msg}) :}$ *The encryption algorithm takes as input the secret key* $\mathsf{sk}$*, and a message* $\mathsf{msg} \in \mathcal{M}$*, and outputs a ciphertext* $\mathsf{ct}$*.*

$\underline{\mathsf{msg} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) :}$ *The decryption algorithm takes as input the secret key* $\mathsf{sk}$ *and the ciphertext* $\mathsf{ct}$*, and outputs a message* $\mathsf{msg}$*.*

*Furthermore, the algorithms must satisfy the following correctness and security properties.*

**Correctness**: *For every message* $\mathsf{msg} \in \mathcal{M}$*, the following holds:*

$$\Pr\left[\mathsf{msg} = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) : \begin{matrix} \mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathsf{msg}) \end{matrix}\right] = 1$$

**IND-CPA security**: *For every PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all* $\lambda \in \mathbb{N}$*, the following holds:*

$$\left|\Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{SKE}}(\lambda, 0) = 1] - \Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{SKE}}(\lambda, 1) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security game is defined as follows:*

$\underline{\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{SKE}}(\lambda, \beta) :}$

1. $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$
2. $(\mathsf{msg}_0, \mathsf{msg}_1) \leftarrow \mathcal{A}(1^\lambda)$
3. $\mathsf{ct}_\beta = \mathsf{Enc}(\mathsf{sk}, \mathsf{msg}_\beta)$
4. ***return*** $\beta' \leftarrow \mathcal{A}(\mathsf{ct}_\beta)$

**Additional IND-CPA security definition.** We consider an additional security definition for SKE, called weak IND$-CPA (wIND$-CPA) security.

**Definition 8** *Let $\Pi_{\mathsf{SKE}}$ be an SKE with ciphertext space denoted by $\mathcal{C}$. We call $\Pi_{\mathsf{SKE}}$ wIND$-CPA secure if for every PPT adversary $\mathcal{A}$ it holds that*

$$\left|\Pr[\mathsf{GwIND\$\text{-}CPA}^{\mathcal{A}}_{\mathsf{SKE}}(\lambda,1)=1]-\Pr[\mathsf{GwIND\$\text{-}CPA}^{\mathcal{A}}_{\mathsf{SKE}}(\lambda,0)=1]\right|\leq \mathsf{negl}(\lambda)$$

*where the security game is defined as follows:*

$\underline{\mathsf{GwIND\$\text{-}CPA}^{\mathcal{A}}_{\mathsf{SKE}}(\lambda,\beta):}$

1. $\mathsf{sk}\leftarrow\mathsf{Gen}(1^\lambda)$
2. $\mathsf{msg}\leftarrow\mathcal{A}(1^\lambda)$
3. $\mathsf{ct}_0\leftarrow\mathsf{Enc}(\mathsf{sk},\mathsf{msg})$
4. $\mathsf{ct}_1\leftarrow\mathcal{C}$
5. ***return*** $\beta'\leftarrow\mathcal{A}(\mathsf{ct}_\beta)$

## 3.5 Public Key Encryption

We first recall the standard syntax of PKE which we require to have indistinguishable encryptions under the chosen plaintext attack (IND-CPA). We note that the scheme may also be defined with respect to a global setup $\mathsf{gp}\leftarrow\mathsf{Setup}(1^\lambda)$ generating parameters $\mathsf{gp}$ used by $\mathsf{Gen}$.

**Definition 9 (PKE)** *We say that a triplet $\Pi_{\mathsf{PKE}}=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ of PPT algorithms with a message space $\mathcal{M}$ is an IND-CPA secure public key encryption (PKE) scheme if the following holds:*

$\underline{(\mathsf{pk},\mathsf{sk})\leftarrow\mathsf{Gen}(1^\lambda):}$ *The key generation algorithm takes as input the security parameter $1^\lambda$, and outputs a public key $\mathsf{pk}$ and a secret key $\mathsf{sk}$.*

$\underline{\mathsf{ct}\leftarrow\mathsf{Enc}(\mathsf{pk},\mathsf{msg}):}$ *The encryption algorithm takes as input the public key $\mathsf{pk}$, and a message $\mathsf{msg}\in\mathcal{M}$, and outputs a ciphertext $\mathsf{ct}$.*

$\underline{\mathsf{msg}\leftarrow\mathsf{Dec}(\mathsf{sk},\mathsf{ct}):}$ *The decryption algorithm takes as input the secret key $\mathsf{sk}$ and a ciphertext $\mathsf{ct}$, and outputs a message $\mathsf{msg}$.*

*Furthermore, the algorithms must satisfy the following correctness and security properties.*

**Correctness**: *For every message $\mathsf{msg}\in\mathcal{M}$, the following holds:*

$$\Pr\left[\mathsf{msg}=\mathsf{Dec}(\mathsf{sk},\mathsf{ct}):\begin{matrix}(\mathsf{pk},\mathsf{sk})\leftarrow\mathsf{Gen}(1^\lambda)\\ \mathsf{ct}\leftarrow\mathsf{Enc}(\mathsf{pk},\mathsf{msg})\end{matrix}\right]=1$$

**IND-CPA security**: *For every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda\in\mathbb{N}$, the following holds:*

$$\left|\Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda,0)=1]-\Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda,1)=1]\right|\leq\mathsf{negl}(\lambda)$$

*where the security game is defined as follows:*

$\underline{\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda,\beta):}$

1. $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
2. $(\mathsf{msg}_0,\mathsf{msg}_1) \leftarrow \mathcal{A}(\mathsf{pk})$
3. $\mathsf{ct}_\beta = \mathsf{Enc}(\mathsf{pk},\mathsf{msg}_\beta)$
4. ***return*** $\beta' \leftarrow \mathcal{A}(\mathsf{ct}_\beta)$

**Additional IND-CPA security definition.** We consider an additional security definition for PKE, called IND$-CPA security.

**Definition 10** *Let $\Pi_{\mathsf{PKE}}$ be a PKE with ciphertext space denoted by $\mathcal{C}$. We call* PKE *IND$-CPA secure if for every PPT adversary $\mathcal{A}$ it holds that*

$$\left|\Pr[\mathsf{GIND\$\text{-}CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda,1)=1] - \Pr[\mathsf{GIND\$\text{-}CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda,0)=1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security game is defined as follows:*

$\underline{\mathsf{GIND\$\text{-}CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda,\beta):}$

1. $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
2. $\mathsf{msg} \leftarrow \mathcal{A}(\mathsf{pk})$
3. $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{pk},\mathsf{msg})$
4. $\mathsf{ct}_1 \leftarrow \mathcal{C}$
5. ***return*** $\beta' \leftarrow \mathcal{A}(\mathsf{ct}_\beta)$

## 3.6 Tag-Based Public Key Encryption

We recall the definitions for tag-based encryption which is used as a building block for our sIND-CCA secure anamorphic encryption scheme in Figure 4.

**Definition 11 (TBE)** *A tag-based public key encryption (TBE) scheme with message space $\mathcal{M}$ and tag space $\mathcal{T}$ is a triplet $\Pi_{\mathsf{TBE}} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ of PPT algorithms such that:*

$\underline{(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda):}$ *The key generation algorithm takes as input the security parameter $1^\lambda$, and outputs a public key* pk *and a secret key* sk*.*

$\underline{\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk},\mathsf{tag},\mathsf{msg}):}$ *The encryption algorithm takes as input the public key* pk*, a* $\mathsf{tag} \in \mathcal{T}$*, and a message* $\mathsf{msg} \in \mathcal{M}$*, and outputs a ciphertext* ct*.*

$\underline{\mathsf{msg} \leftarrow \mathsf{Dec}(\mathsf{sk},\mathsf{tag},\mathsf{ct}):}$ *The decryption algorithm takes as input the secret key* sk*, a* $\mathsf{tag} \in \mathcal{T}$*, and a ciphertext* ct*, and outputs a message* msg *or the failure symbol* $\perp$*.*

*We call $\Pi_{\mathsf{TBE}}$ correct, if for every message* $\mathsf{msg} \in \mathcal{M}$*, and tag* $\mathsf{tag} \in \mathcal{T}$*, the following holds:*

$$\Pr\left[\mathsf{msg} = \mathsf{Dec}(\mathsf{sk},\mathsf{tag},\mathsf{ct}) : \begin{matrix}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk},\mathsf{tag},\mathsf{msg})\end{matrix}\right] = 1$$

**Definition 12 (IND-CCA Security for TBE)** *A TBE* $\Pi_{\mathsf{TBE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is called IND-CCA secure if for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all* $\lambda \in \mathbb{N}$*, the following holds:*

$$\left|\Pr[\mathsf{GIND\text{-}CCA}^{\mathcal{A}}_{\mathsf{TBE}}(\lambda, 0) = 1] - \Pr[\mathsf{GIND\text{-}CCA}^{\mathcal{A}}_{\mathsf{TBE}}(\lambda, 1) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security game is defined as follows:*

$\underline{\mathsf{GIND\text{-}CCA}^{\mathcal{A}}_{\mathsf{TBE}}(\lambda, \beta) :}$

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
2. $(\mathsf{msg}_0, \mathsf{msg}_1, \mathsf{tag}^*) \leftarrow \mathcal{A}^{\mathsf{ODec}(\cdot,\cdot)}(\mathsf{pk})$
3. $\mathsf{ct}_\beta \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{tag}, \mathsf{msg}_\beta)$
4. ***return*** $\beta' \leftarrow \mathcal{A}^{\mathsf{ODec}(\cdot,\cdot)}(\mathsf{ct}_\beta)$

   *where* $\mathsf{ODec}(\mathsf{tag}, \mathsf{ct}) = \begin{cases} \bot, & \text{if } (\mathsf{tag}, \mathsf{ct}) = (\mathsf{tag}^*, \mathsf{ct}_\beta) \\ \mathsf{Dec}(\mathsf{sk}, \mathsf{tag}, \mathsf{ct}), & \text{else} \end{cases}$

**Definition 13 (IND$-CPA Security for TBE)** *Let* $\Pi_{\mathsf{TBE}}$ *be a TBE with ciphertext space denoted by* $\mathcal{C}$*. We call* $\Pi_{\mathsf{TBE}}$ *IND$-CPA secure if for every PPT adversary* $\mathcal{A}$ *it holds that*

$$\left|\Pr[\mathsf{GIND\$\text{-}CPA}^{\mathcal{A}}_{\mathsf{TBE}}(\lambda, 1) = 1] - \Pr[\mathsf{GIND\$\text{-}CPA}^{\mathcal{A}}_{\mathsf{TBE}}(\lambda, 0) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security game is defined as follows:*

$\underline{\mathsf{GIND\$\text{-}CPA}^{\mathcal{A}}_{\mathsf{TBE}}(\lambda, \beta) :}$

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
2. $(\mathsf{msg}, \mathsf{tag}) \leftarrow \mathcal{A}(\mathsf{pk})$
3. $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{tag}, \mathsf{msg})$
4. $\mathsf{ct}_1 \leftarrow \mathcal{C}$
5. ***return*** $\beta' \leftarrow \mathcal{A}(\mathsf{ct}_\beta)$

## 3.7 Identity-Based Encryption

**Definition 14 (IBE)** *We say that a triplet* $\Pi_{\mathsf{IBE}} = (\mathsf{Setup}, \mathsf{Enc}, , \mathsf{KeyGen}, \mathsf{Dec})$ *of PPT algorithms with a message space* $\mathcal{M}$*, identity space* $\mathcal{ID}$ *is an IND-CPA secure identity-based encryption (IBE) scheme if the following holds:*

$\underline{(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) :}$ *The setup algorithm takes as input the security parameter* $1^\lambda$*, and outputs a master public key* $\mathsf{mpk}$ *and a master secret key* $\mathsf{msk}$*.*

$\underline{\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) :}$ *The key generation algorithm takes as input the master secret key* $\mathsf{msk}$ *and an identity* $\mathsf{id} \in \mathcal{ID}$*, and outputs a secret key* $\mathsf{sk}_{\mathsf{id}}$*.*

$\underline{\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{msg}) :}$ *The encryption algorithm takes as input the master public key* $\mathsf{mpk}$*, an identity* $\mathsf{id} \in \mathcal{ID}$ *and a message* $\mathsf{msg} \in \mathcal{M}$*, and outputs a ciphertext* $\mathsf{ct}$*.*

$\underline{\mathsf{msg} \leftarrow \mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) :}$ *The decryption algorithm takes as input a secret key* $\mathsf{sk}_{\mathsf{id}}$ *and a ciphertext* $\mathsf{ct}$*, and outputs a message* $\mathsf{msg}$*.*

*Furthermore, the algorithms must satisfy the following correctness and security properties.*

**Correctness**: *For every message* $\mathsf{msg} \in \mathcal{M}$ *and identity* $\mathsf{id} \in \mathcal{ID}$*, the following holds:*

$$\Pr\left[\mathsf{msg} = \mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) : \begin{array}{c}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{msg})\end{array}\right] = 1$$

**IND-CPA security**: *For every PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all* $\lambda \in \mathbb{N}$*, the following holds:*

$$\left|\Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBE}}(\lambda, 0) = 1] - \Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBE}}(\lambda, 1) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security game is defined as follows:*

$\underline{\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBE}}(\lambda, \beta) :}$

1. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{id}^*, \mathsf{msg}_0, \mathsf{msg}_1) \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot)}(\mathsf{mpk})$
3. $\mathsf{ct}_\beta = \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, \mathsf{msg}_\beta)$
4. ***return*** $\beta' \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot)}(\mathsf{ct}_\beta)$

   *where* $\mathsf{OKeyGen}(\mathsf{id}) = \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$

   *and* $\mathsf{id}^*$ *is not queried to* $\mathsf{OKeyGen}(\cdot)$.

**Additional IND-CPA security definition.** We consider an additional security definition for IBE, called IND$-CPA security.

**Definition 15** *Let* $\Pi_{\mathsf{IBE}}$ *be an IBE with ciphertext space denoted by* $\mathcal{C}$*. We call* IBE *IND$-CPA secure if for every PPT adversary* $\mathcal{A}$ *it holds that*

$$\left|\Pr[\mathsf{GIND\$\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBE}}(\lambda, 0) = 1] - \Pr[\mathsf{GIND\$\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBE}}(\lambda, 1) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security game is defined as follows:*

$\underline{\mathsf{GIND\$\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBE}}(\lambda, \beta) :}$

1. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{id}^*, \mathsf{msg}) \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot)}(\mathsf{mpk})$
3. $\mathsf{ct}_0 = \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, \mathsf{msg})$
4. $\mathsf{ct}_1 \leftarrow \mathcal{C}$
5. ***return*** $\beta' \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot)}(\mathsf{ct}_\beta)$

   *where* $\mathsf{OKeyGen}(\mathsf{id}) = \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$

   *and* $\mathsf{id}^*$ *is not queried to* $\mathsf{OKeyGen}(\cdot)$.

## 3.8 Non-Interactive Key Exchange

We recall the definitions for non-interactive key exchange which is used as a building block for our sIND-CCA secure anamorphic encryption scheme in Figure 4.

**Definition 16 (NIKE)** *A non-interactive key-exchange (NIKE) scheme with key space $\mathcal{K}$ is a triplet $\Pi_{\mathsf{NIKE}} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{SharedKey})$ of PPT algorithms such that:*

$\underline{\mathsf{nikepar} \leftarrow \mathsf{Setup}(1^\lambda):}$ *The setup algorithm takes as input the security parameter $1^\lambda$, and outputs a public parameters* nikepar *for the key exchange.*

$\underline{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{nikepar}):}$ *The generation algorithm takes as input the public parameters* nikepar *and generates a key pair* $(\mathsf{pk}, \mathsf{sk})$ *as output.*

$\underline{\mathsf{k}_{i,j} \leftarrow \mathsf{SharedKey}(\mathsf{pk}_i, \mathsf{sk}_j):}$ *The shared key generation algorithm takes as input a public key* $\mathsf{pk}_i$ *and a secret key* $\mathsf{sk}_i$*, and outputs a shared key* $\mathsf{k}_{i,j} \in \mathcal{K}$.

*We call $\Pi_{\mathsf{NIKE}}$ correct, if for every $\lambda \in \mathbb{N}$ holds:*

$$\Pr\left[\mathsf{SharedKey}(\mathsf{pk}_i, \mathsf{sk}_j) = \mathsf{SharedKey}(\mathsf{pk}_j, \mathsf{sk}_i) : \begin{array}{r} \mathsf{nikepar} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(\mathsf{nikepar}) \\ (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{Gen}(\mathsf{nikepar}) \end{array}\right] = 1$$

**Definition 17 (Security for NIKE)** *Let $\Pi_{\mathsf{NIKE}}$ be a NIKE with (finite) key space denoted by $\mathcal{K}$. We call $\Pi_{\mathsf{NIKE}}$ secure (CKS-light security in the honest key registration setting [FHKP13]) if for every PPT adversary $\mathcal{A}$ it holds that*

$$\left|\Pr[\mathsf{GCKS\text{-}LIGHT}^{\mathcal{A}}_{\mathsf{NIKE}}(\lambda, 1) = 1] - \Pr[\mathsf{GCKS\text{-}LIGHT}^{\mathcal{A}}_{\mathsf{NIKE}}(\lambda, 0) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security game is defined as follows:*

$\underline{\mathsf{GCKS\text{-}LIGHT}^{\mathcal{A}}_{\mathsf{NIKE}}(\lambda, \beta):}$

1. $\mathsf{nikepar} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(\mathsf{nikepar})$
3. $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{Gen}(\mathsf{nikepar})$
4. $\mathsf{k}_0 \leftarrow \mathsf{SharedKey}(\mathsf{pk}_i, \mathsf{sk}_j)$
5. $\mathsf{k}_1 \leftarrow \mathcal{K}$
6. ***return*** $\beta' \leftarrow \mathcal{A}(\mathsf{nikepar}, \mathsf{pk}_i, \mathsf{pk}_j, \mathsf{k}_\beta)$

Apart from the standard security property for NIKE as formalized before, we also need that public keys generated by the NIKE scheme look random. This is, e.g., satisfied by the Diffie-Hellman Key Exchange.

**Definition 18 (NIKE with Pseudorandom Public Keys)** *Let $\Pi_{\mathsf{NIKE}}$ be a NIKE with (finite) public key space (depending on $\lambda$) denoted by $\mathcal{PK}$. Then we say that $\Pi_{\mathsf{NIKE}}$ generates pseudrandom public keys if for all PPT adversaries $\mathcal{A}$, $\lambda \in \mathbb{N}$, and parameters $\mathsf{nikepar} \leftarrow \mathsf{Setup}(1^\lambda)$ it holds that*

$$\left|\Pr[\mathcal{A}^{\mathsf{GenPK}(\mathsf{nikepar})}(\mathsf{nikepar}) = 1] - \Pr[\mathcal{A}^{\mathsf{RandPK}(\mathsf{nikepar})}(\mathsf{nikepar}) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where* GenPK *is an oracle which when triggered invokes* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{nikepar})$ *and returns only the first component, and* RandPK *is an oracle which returns* $\mathsf{pk} \leftarrow \mathcal{PK}$ *when triggered.*

## 3.9 Signature Scheme

We now recall the definition of a basic signature scheme and note that the scheme may also be defined with respect to a global setup with parameters gp.

**Definition 19 (Signature Scheme)** *We say that a triplet* $\Pi_{\mathsf{s}} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *of PPT algorithms with a message space* $\mathcal{M}$ *is a signature scheme if the following holds:*

$(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ : *The key generation algorithm* Gen *takes as input the security parameter* $1^\lambda$*, and outputs a signing key* sk *and a verification key* vk.

$\mathsf{sig} \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{msg})$ : *The signing algorithm* Sig *takes as input the signing key* sk*, and a message* $\mathsf{msg} \in \mathcal{M}$ *and outputs a signature* sig.

$1/0 \leftarrow \mathsf{Ver}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$ : *The verification algorithm takes input the verification key* vk*, the message* msg *and the signature* sig*, and outputs* 1 *or* 0.

*Furthermore, the algorithms must satisfy the following correctness and security properties.*

**Correctness**: *For every message* $\mathsf{msg} \in \mathcal{M}$*, the following holds:*

$$\Pr\left[\mathsf{Ver}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) = 1 : \begin{matrix}(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \mathsf{sig} \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{msg})\end{matrix}\right] = 1$$

**Security**: *The signature scheme* $\Pi_{\mathsf{SS}}$ *is secure against chosen-message attacks if for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all* $\lambda \in \mathbb{N}$*, the following holds:*

$$\left|\Pr[\mathsf{GSig\text{-}CMA}^{\mathcal{A}}_{\Pi_{\mathsf{SS}}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security game is defined as follows:*

$\mathsf{GSig\text{-}CMA}^{\mathcal{A}}_{\Pi_{\mathsf{SS}}}(\lambda)$ :

1. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
2. $(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}^{\mathsf{OSig}(\mathsf{sk}, \cdot)}(\mathsf{vk})$

   *where* $\mathsf{OSig}(\mathsf{sk}, \mathsf{msg}) = (\mathsf{msg}, \mathsf{sig})$
3. **if** $\mathsf{Ver}(\mathsf{vk}, \mathsf{msg}^*, \mathsf{sig}^*) = 1$ *and*

   $(\mathsf{msg}^*, \mathsf{sig}^*)$ *has not returned by* OSig

   ***return*** 1*;*

   **else return** 0

## 3.10 Asymmetric Anamorphic Encryption

We now present the relevant notions for AAE from the literature. We follow the current line of works on *receiver* anamorphic encryption introduced in [PPY22], and subsequently generalized in [BGH+24, CGM24a, KPP+23a]. Given our core motivation is to realize covert channels using an asymmetric variant, we follow the foundational work on AAE in [PPY24a, BGH+24].

At a high-level, an AAE is a triplet $\Pi = (\mathsf{aGen}, \mathsf{aEnc}, \mathsf{aDec})$ associated with a regular PKE $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ which operates in either of the two modes: *regular* and *anamorphic*. The regular mode is basically the ordinary encryption scheme $\Pi_{\mathsf{PKE}}$ where the receiver generates a key-pair $(\mathsf{pk}, \mathsf{sk})$ using Gen, and hands over the secret key sk to the dictator, who can then decrypt any ciphertext ct meant for the receiver. In the anamorphic mode, the receiver runs aGen and generates anamorphic key-pair $(\mathsf{apk}, \mathsf{ask})$ along with encryption and decryption *double* keys $(\mathsf{dk}, \mathsf{tk})$. When a sender wants to send an anamorphic message amsg, she chooses an innocuous-looking message msg and runs aEnc, such that the output ciphertext act decrypts to

$\mathsf{amsg}$ using $\mathsf{tk}$ on $\mathsf{aDec}$, and $\mathsf{msg}$ using $\mathsf{ask}$ on $\mathsf{Dec}$. Additionally, $\mathsf{aDec}$ might require to identify the existence of non-anamorphic messages. This is captured by a notion called *robustness* proposed in [BGH+24], which essentially requires $\mathsf{aDec}$ to return $\perp$ when decrypting a regular message $\mathsf{msg}$ encrypted with $\mathsf{Enc}$, with overwhelming probability. *Anamorphic security* essentially requires that a PPT dictator should not be able to tell apart the output distributions of regular and anamorphic ciphertexts, along with the corresponding key pairs. *IND-CPA security* for AAE is a security property with respect to the group of peers using the anamorphic channel. Here, an adversary given $(\mathsf{apk}, \mathsf{dk})$ must not be able to derive information about $\mathsf{amsg}$ from a given ciphertext. Below, we also define several (obvious) security notions beyond IND-CPA.

**Definition 20 (AAE [CGM24a] [PPY24b])** *An asymmetric anamorphic encryption (AAE) scheme with anamorphic message space $\widehat{\mathcal{M}}$ associated with a public key encryption scheme $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with a regular message space $\mathcal{M}$ is a triplet $\Pi = (\mathsf{aGen}, \mathsf{aEnc}, \mathsf{aDec})$ of PPT algorithms defined as follows:*

$(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda)$ : *The anamorphic key generation algorithm* $\mathsf{aGen}$ *takes as input the security parameter $1^\lambda$, and outputs an anamorphic public key* $\mathsf{apk}$*, an anamorphic secret key* $\mathsf{ask}$*, an encryption double key* $\mathsf{dk}$ *and a decryption double key* $\mathsf{tk}$*.*

$\mathsf{act} \leftarrow \mathsf{aEnc}(\mathsf{apk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$ : *The anamorphic encryption algorithm* $\mathsf{aEnc}$ *takes as input the anamorphic public key* $\mathsf{apk}$*, the encryption double key* $\mathsf{dk}$*, a regular message* $\mathsf{msg} \in \mathcal{M}$ *and an anamorphic message* $\mathsf{amsg} \in \widehat{\mathcal{M}}$*, and outputs an anamorphic ciphertext* $\mathsf{act}$*.*

$\mathsf{amsg} \leftarrow \mathsf{aDec}(\mathsf{tk}, \mathsf{act})$ : *The anamorphic decryption algorithm* $\mathsf{aDec}$ *takes as input the secret decryption double key* $\mathsf{tk}$ *and the anamorphic ciphertext* $\mathsf{act}$*, and outputs an anamorphic message* $\mathsf{amsg}$ *or a failure symbol.*

*Furthermore, the algorithms must satisfy the following properties.*

**Correctness**: *We call $\Pi$ correct if for every pair of messages $(\mathsf{msg}, \mathsf{amsg}) \in \mathcal{M} \times \widehat{\mathcal{M}}$, it holds that*

$$\Pr\left[\mathsf{amsg} \neq \mathsf{aDec}(\mathsf{tk}, \mathsf{act}) : \begin{array}{c}(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda) \\ \mathsf{act} \leftarrow \mathsf{aEnc}(\mathsf{apk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})\end{array}\right] \leq \mathsf{negl}(\lambda)$$

*where the probability is taken over the random coins tosses of* $\mathsf{aGen}$ *and* $\mathsf{aEnc}$*.*

**Anamorphic security**: *$\Pi$ satisfies anamorphic security if for every PPT dictator $\mathcal{D}$ it holds that*

$$\left|\Pr[\mathsf{GReal}^{\mathcal{D}}_{\mathsf{PKE}}(\lambda) = 1] - \Pr[\mathsf{GAnam}^{\mathcal{D}}_{\Pi}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security games are defined as follows:*

| $\mathsf{GReal}^{\mathcal{D}}_{\mathsf{PKE}}(\lambda)$ : | $\mathsf{GAnam}^{\mathcal{D}}_{\Pi}(\lambda)$ : |
|---|---|
| 1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ | 1. $(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda)$ |
| 2. ***return*** $\mathcal{D}^{\mathsf{OEnc}(\cdot,\cdot)}(\mathsf{pk}, \mathsf{sk})$ *where* | 2. ***return*** $\mathcal{D}^{\mathsf{OaEnc}(\cdot,\cdot)}(\mathsf{apk}, \mathsf{ask})$ *where* |
| $\mathsf{OEnc}(\mathsf{msg}, \mathsf{amsg}) = \mathsf{Enc}(\mathsf{pk}, \mathsf{msg})$ | $\mathsf{OaEnc}(\mathsf{msg}, \mathsf{amsg}) = \mathsf{aEnc}(\mathsf{apk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$ |

**Fully asymmetric IND-CPA security**: *$\Pi$ satisfies fully asymmetric IND-CPA security if for every PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$, the following holds:*

$$\left|\Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\Pi}(\lambda, 0) = 1] - \Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\Pi}(\lambda, 1) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security game is defined as follows:*

$\underline{\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\Pi}(\lambda,\beta):}$

1. $(\mathsf{apk},\mathsf{ask},\mathsf{dk},\mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda)$
2. $(\mathsf{msg}_0,\mathsf{msg}_1,\mathsf{amsg}_0,\mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk},\mathsf{dk})$
3. $\mathsf{act}_\beta \leftarrow \mathsf{aEnc}(\mathsf{apk},\mathsf{dk},\mathsf{msg}_\beta,\mathsf{amsg}_\beta)$
4. ***return*** $\mathcal{A}(\mathsf{act}_\beta)$

Additionally, we consider new but reasonable security notions for $\Pi_{\mathsf{AAE}}$. [4]

**Strong IND-CPA security**: $\Pi_{\mathsf{AAE}}$ satisfies strong IND-CPA (sIND-CPA) security if for every PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\left|\Pr[\mathsf{GsIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda,0)=1] - \Pr[\mathsf{GsIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda,1)=1]\right| \leq \mathsf{negl}(\lambda)$$

where the security game is defined as follows:

$\underline{\mathsf{GsIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda,\beta):}$

1. $(\mathsf{apk},\mathsf{ask},\mathsf{dk},\mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda)$
2. $(\mathsf{msg},\mathsf{amsg}_0,\mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk},\mathsf{ask},\mathsf{dk})$
3. $\mathsf{act}_\beta \leftarrow \mathsf{aEnc}(\mathsf{apk},\mathsf{dk},\mathsf{msg},\mathsf{amsg}_\beta)$
4. **return** $\mathcal{A}(\mathsf{act}_\beta)$

**sIND-RCCA security**: $\Pi_{\mathsf{AAE}}$ satisfies sIND-RCCA security if for every PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\left|\Pr[\mathsf{GsIND\text{-}RCCA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda,0)=1] - \Pr[\mathsf{GsIND\text{-}RCCA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda,1)=1]\right| \leq \mathsf{negl}(\lambda)$$

where the security game is defined as follows:

$\underline{\mathsf{GsIND\text{-}RCCA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda,\beta):}$

1. $(\mathsf{apk},\mathsf{ask},\mathsf{dk},\mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda)$
2. $(\mathbf{msg},\mathsf{amsg}_0,\mathsf{amsg}_1) \leftarrow \mathcal{A}^{\mathsf{OaDec}(\cdot)}(\mathsf{apk},\mathsf{ask},\mathsf{dk})$
3. $\mathbf{act}_\beta \leftarrow \mathsf{aEnc}(\mathsf{apk},\mathsf{dk},\mathbf{msg},\mathsf{amsg}_\beta)$
4. **return** $\mathcal{A}^{\mathsf{OaDec}(\cdot)}(\mathbf{act}_\beta)$

   where $\mathsf{OaDec}(\mathbf{act}) = \mathsf{aDec}(\mathsf{tk},\mathbf{act})$ and

   $\mathsf{OaDec}(\mathbf{act})$ outputs $\perp$ if $\mathsf{aDec}(\mathsf{tk},\mathbf{act}) = \mathsf{amsg}_0$ or $\mathsf{amsg}_1$.

[4] Similar security notions have been concurrently and independently proposed by Choi, Collins, Liu and Zikas in [CCLZ25], however without any concrete instantiations.

**sIND-RCCA security**: $\Pi_{\mathsf{AAE}}$ satisfies sIND-RCCA security if for every PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\left|\Pr[\mathsf{GsIND\text{-}RCCA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda,0)=1] - \Pr[\mathsf{GsIND\text{-}RCCA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda,1)=1]\right| \leq \mathsf{negl}(\lambda)$$

where the security game is defined as follows:

$\underline{\mathsf{GsIND\text{-}RCCA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda,\beta):}$

1. $(\mathsf{apk},\mathsf{ask},\mathsf{dk},\mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda)$
2. $(\mathbf{msg},\mathsf{amsg}_0,\mathsf{amsg}_1) \leftarrow \mathcal{A}^{\mathsf{OaDec}(\cdot)}(\mathsf{apk},\mathsf{ask},\mathsf{dk})$
3. $\mathbf{act}_\beta \leftarrow \mathsf{aEnc}(\mathsf{apk},\mathsf{dk},\mathbf{msg},\mathsf{amsg}_\beta)$
4. **return** $\mathcal{A}^{\mathsf{OaDec}(\cdot)}(\mathbf{act}_\beta)$

   where $\mathsf{OaDec}(\mathbf{act}) = \mathsf{aDec}(\mathsf{tk},\mathbf{act})$ and

   $\mathsf{OaDec}(\mathbf{act})$ outputs $\perp$ if $\mathsf{aDec}(\mathsf{tk},\mathbf{act}) = \mathsf{amsg}_0$ or $\mathsf{amsg}_1$.

**sIND-CCA security**: $\Pi_{\mathsf{AAE}}$ satisfies sIND-CCA security if for every PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\left|\Pr[\mathsf{GIND\text{-}CCA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda,0)=1] - \Pr[\mathsf{GIND\text{-}CCA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda,1)=1]\right| \leq \mathsf{negl}(\lambda)$$

where the security game is defined as follows:

$\underline{\mathsf{GIND\text{-}CCA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda,\beta):}$

1. $(\mathsf{apk},\mathsf{ask},\mathsf{dk},\mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda)$
2. $(\mathsf{msg},\mathsf{amsg}_0,\mathsf{amsg}_1) \leftarrow \mathcal{A}^{\mathsf{OaDec}(\cdot)}(\mathsf{apk},\mathsf{ask},\mathsf{dk})$
3. $\mathsf{act}_\beta \leftarrow \mathsf{aEnc}(\mathsf{apk},\mathsf{dk},\mathsf{msg},\mathsf{amsg}_\beta)$
4. **return** $\mathcal{A}^{\mathsf{OaDec}(\cdot)}(\mathsf{act}_\beta)$

   where $\mathsf{OaDec}(\mathsf{act}) = \mathsf{aDec}(\mathsf{tk},\mathsf{act})$ and

   $\mathsf{OaDec}(\mathsf{act})$ outputs $\perp$ if $\mathsf{act} = \mathsf{act}_\beta$.

# 4 $\mu$-Message Asymmetric Anamorphic Encryption Extension

We present concrete realizations of AAE extensions based on the ElGamal [ElG86] (Section 4.1) and the Dual-Regev [GPV08] (Section 4.2) cryptosystems as well as the Fujisaki-Okamoto transformation [FO13] (Section 4.3).

**Definition 21 ($\mu$-Message AAE Extension)** *For $\mu \in \mathbb{N}$, an asymmetric anamorphic encryption extension ($\mu$-AAEE) scheme with anamorphic message space $\widehat{\mathcal{M}}$ associated with a public key encryption scheme $\Pi_{\mathsf{PKE}} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ with a regular message space $\mathcal{M}$ is a triplet $\Pi_{\mathsf{AAE}} = (\mathsf{aGen},\mathsf{aEnc},\mathsf{aDec})$ of PPT algorithms defined as follows:*

$(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda)$ : *The anamorphic key generation algorithm* aGen *takes as input the security parameter* $1^\lambda$*, and outputs the anamorphic public key* apk*, anamorphic secret key* ask*, an encryption double key* dk *and a decryption double key* tk*.*

$\mathbf{act} \leftarrow \mathsf{aEnc}(\mathsf{apk}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg})$ : *The anamorphic encryption algorithm* aEnc *takes as input the anamorphic public key* apk*, the encryption double key* dk*, a regular message vector* $\mathbf{msg} \in \mathcal{M}^\mu$*, and an anamorphic message* $\mathsf{amsg} \in \widehat{\mathcal{M}}$*, and outputs an anamorphic ciphertext vector* $\mathbf{act}$*.*

$\mathsf{amsg} \leftarrow \mathsf{aDec}(\mathsf{tk}, \mathbf{act})$ : *The anamorphic decryption algorithm* aDec *takes as input the decryption double key* tk *and the anamorphic ciphertext* $\mathbf{act}$*, and outputs an anamorphic message* amsg *or a failure symbol* $\perp \notin \hat{\mathcal{M}}$*.*

**Anamorphic security**: $\Pi_{\mathsf{AAE}}$ *satisfies anamorphic security if for every PPT dictator* $\mathcal{D}$ *it holds that*

$$\left|\Pr[\mathsf{GReal}^{\mathcal{D}}_{\mathsf{PKE}}(\lambda) = 1] - \Pr[\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{AAE}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security games are defined as follows:*

$\mathsf{GReal}^{\mathcal{D}}_{\mathsf{PKE}}(\lambda)$ :

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
2. ***return*** $\mathcal{D}^{\mathsf{OEnc}(\cdot,\cdot)}(\mathsf{pk}, \mathsf{sk})$

*where* $\mathsf{OEnc}(\mathbf{msg}, \mathsf{amsg}) = \mathbf{ct}$*,*

$\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}, \mathsf{msg}_i)$*,*

$\mathbf{msg} = \{\mathsf{msg}_i\}_{i\in[\mu]}, \mathbf{ct} = \{\mathsf{ct}_i\}_{i\in[\mu]}$

$\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{AAE}}(\lambda)$ :

1. $(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda)$
2. ***return*** $\mathcal{D}^{\mathsf{OaEnc}(\cdot,\cdot)}(\mathsf{apk}, \mathsf{ask})$

*where* $\mathsf{OaEnc}(\mathbf{msg}, \mathsf{amsg}) =$

$\mathsf{aEnc}(\mathsf{apk}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg})$

**Fully asymmetric IND-CPA security**: $\Pi_{\mathsf{AAE}}$ *satisfies fully asymmetric IND-CPA security if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all* $\lambda \in \mathbb{N}$*, the following holds:*

$$\left|\Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda, 0) = 1] - \Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda, 1) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security game is defined as follows:*

$\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda, \beta)$ :

1. $(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda)$
2. $(\mathbf{msg}_0, \mathbf{msg}_1, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk}, \mathsf{dk})$
3. $\mathbf{act}_\beta \leftarrow \mathsf{aEnc}(\mathsf{apk}, \mathsf{dk}, \mathbf{msg}_\beta, \mathsf{amsg}_\beta)$
4. ***return*** $\mathcal{A}(\mathbf{act}_\beta)$

**Strong IND-CPA and strong IND-(R)CCA security**: In the following, we introduce new but reasonable security notions for $\Pi_{\mathsf{AAE}}$. [5] To guarantee confidentiality also with respect to a dictator who became aware of the covert channel (obtained dk) and wants to silently surveil covert communication, we consider IND-CPA/CCA-like definitions (such as the one above) which additionally provide ask as input to $\mathcal{A}$. Note

[5] Similar security notions have been concurrently and independently proposed by Choi, Collins, Liu and Zikas in [CCLZ25], however without any concrete instantiations.

that, in this case, letting the adversary choose two regular messages $\mathbf{msg}_0$ and $\mathbf{msg}_1$ would lead to an adversary which trivially wins the game.

$\Pi_{\mathsf{AAE}}$ satisfies strong IND-CPA (sIND-CPA) security if for every PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\left|\Pr[\mathsf{GsIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda, 0) = 1] - \Pr[\mathsf{GsIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda, 1) = 1]\right| \leq \mathsf{negl}(\lambda)$$

where the security game is defined as follows:

$\underline{\mathsf{GsIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda, \beta) :}$

1. $(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda)$
2. $(\mathbf{msg}, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk}, \mathsf{ask}, \mathsf{dk})$
3. $\mathbf{act}_\beta \leftarrow \mathsf{aEnc}(\mathsf{apk}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg}_\beta)$
4. **return** $\mathcal{A}(\mathbf{act}_\beta)$

We call $\Pi_{\mathsf{AAE}}$ strong IND-CCA (sIND-CCA) secure if the advantage of $\mathcal{A}$ in $\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AAE}}$ is negligible when it is additionally given access to an $\mathsf{aDec}(\mathsf{tk}, \cdot)$-oracle which can be queried with any $\mathbf{act} \neq \mathbf{act}_\beta$. Since sIND-CCA seems to require more complex constructions, we also consider a relaxation in the spirit of Replayable CCA (RCCA) [CKN03] security for PKE which allows for quite simple constructions: RCCA captures a security notion which is identical to CCA except for the added ability of the attacker to generate new ciphertexts that decrypt to the same plaintexts as previously seen ciphertexts (i.e., it allows for re-randomization). More precisely, our sIND-RCCA game coincides with the sIND-CCA game, except that the $\mathsf{aDec}(\mathsf{tk}, \cdot)$-oracle rejects for a given $\mathbf{act}$ if $\mathsf{aDec}(\mathsf{tk}, \mathbf{act})$ results in $\mathsf{amsg}_0$ or $\mathsf{amsg}_1$.

**Remark 1 (Non-adaptive choice of message)** *Our current definitions focus on the non-adaptive setting, where the adversary specifies the regular-message vector in one batch. Since our goal is to minimize $\mu$ (in some of our constructions, $\mu = 2$), it is natural to treat such a small set of messages jointly. We intentionally adopt the non-adaptive model for conceptual clarity, cleaner black-box reductions, and to avoid technical overhead that would obscure the core ideas. Moreover, the model already captures practical scenarios where a small batch of messages (e.g., emails or transactions) is influenced together.*

*That said, adaptive setting is a natural and interesting strengthening, and some of our constructions already appear compatible with adaptive message selection. We also note that defining adaptive security is not entirely straightforward in our general framework, since the current interface takes a full message vector as input and outputs an anamorphic ciphertext vector. Achieving adaptivity may therefore require restricting to a more specialized class of AAE extensions supporting layered or incremental embedding, which we leave as an interesting direction for future work.*

**AAE-compatibility for PKE**: At a high-level, this new notion encompasses two properties. The first property *on-the-fly double key generation* captures the essence of anamorphic extension since the double keys $\mathsf{dk}$ and $\mathsf{tk}$ can simply be derived from regular key-pairs already in use. More precisely, for key pairs $(\mathsf{pk}, \mathsf{sk})$ and $(\mathsf{pk}', \mathsf{sk}')$ generated by $\mathsf{PKE.Gen}$, we set $\mathsf{sk}'$ as $\mathsf{tk}$, while $\mathsf{dk}$ is set as a function of the key pairs. The second property indicates that $\mathsf{aDec}$ is essentially the same as $\mathsf{PKE.Dec}$ under the existence of a publicly known poly-time function $F$. In particular, $F(\mathbf{act})$ produces a valid ciphertext that decrypts to $\mathsf{amsg}$ using $\mathsf{PKE.Dec}$. We call this property *ciphertext extractibility*. Note that the recently proposed simple asymmetric anamorphic construction based on CS-Lite [CGM24a], for instance, does not satisfy the AAE-compatibility notion.

**Definition 22 (AAE-Compatible)** *A public key encryption scheme $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be AAE-compatible for the triplet $\Pi_{\mathsf{AAE}} = (\mathsf{aGen}, \mathsf{aEnc}, \mathsf{aDec})$ if the following holds:*

- *On-the-fly double key generation: A part of output* $(\mathsf{dk}, \mathsf{tk})$ *of* $\mathsf{AAE.aGen}(1^\lambda)$ *is the output of* $\mathsf{PKE.Gen}(1^\lambda)$. *In particular, if* $\mathsf{AAE.aGen}(1^\lambda) = (\mathsf{ask}, \mathsf{apk}, \mathsf{dk}, \mathsf{tk})$ *and* $\mathsf{PKE.Gen}(1^\lambda) = (\mathsf{pk}, \mathsf{sk})$, $\mathsf{PKE.Gen}(1^\lambda) = (\mathsf{pk}', \mathsf{sk}')$ *then* $\mathsf{apk} = \mathsf{pk}, \mathsf{ask} = \mathsf{sk}, \mathsf{tk} = \mathsf{sk}'$ *and there exists an efficiently computable function* $G$ *such that* $\mathsf{dk} = G(\mathsf{pk}, \mathsf{sk}, \mathsf{pk}', \mathsf{sk}')$.
- *Ciphertext extractibility: There exists an efficiently computable function* $F$ *such that* $\mathsf{PKE.Dec}(\mathsf{tk}, F(\mathbf{act})) = \mathsf{amsg}$[6], *where* $\mathbf{act} = \mathsf{AAE.aEnc}(\mathsf{apk}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg})$.

## 4.1 Extension of the ElGamal Encryption Scheme

We now show a concrete AAE construction based on the well-known ElGamal encryption scheme [ElG86]. The ElGamal PKE scheme is a triplet of PPT algorithms $\Pi_{\mathsf{El}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ defined over a cyclic group $\mathbb{G} = \langle g \rangle$ of order $q$. The key generation algorithm $\mathsf{El.Gen}$ on input $1^\lambda$, samples a uniformly random $x$ in $\mathbb{Z}_q$, and returns $(\mathsf{sk} := x, \mathsf{pk} := g^x)$. To encrypt a message $\mathsf{msg} \in \mathcal{M}$, where $\mathcal{M}$ is the group $\mathbb{G}$ itself, the encryption algorithm $\mathsf{El.Enc}$ selects a uniformly random $\kappa$ in $\mathbb{Z}_q$, and computes the ciphertext $\mathsf{ct}$ as $(\mathsf{ct}_1, \mathsf{ct}_2) = (g^\kappa, \mathsf{pk}^\kappa \cdot \mathsf{msg})$. The decryption algorithm $\mathsf{EL.Dec}$ simply recovers $\mathsf{msg}$ by computing $\mathsf{ct}_2 \cdot (\mathsf{ct}_1)^{-\mathsf{sk}}$. We can also consider ElGamal for $\mathcal{M}$ being a polynomially-bounded subspace of $\mathbb{Z}_q$, then $\mathsf{msg}$ can be recovered by computing the DL of $\mathsf{ct}_2 \cdot (\mathsf{ct}_1)^{-\mathsf{sk}}$. It is well-known that ElGamal is IND-CPA secure under the $\mathsf{DDH}$ assumption in $\mathbb{G}$.

Recall from our earlier discussions (Section 2), that our goal is to embed $\mathsf{amsg}$ within $\mu$ ciphertexts. With the ElGamal encryption scheme, we achieve this goal using $\mu = 2$ ciphertexts.

**ElGamal 2-Message AAE.** We begin with an interesting observation that $\mathbf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ (vector with *two* consecutive ciphertexts) precisely contains two uniformly random group elements $\kappa_1, \kappa_2$ (first components of $\mathsf{ct}_1$ and $\mathsf{ct}_2$ respectively). If we could bias these random choices in a way that it covertly embeds $\mathsf{amsg}$, then we would be able to recover $\mathsf{amsg}$ by treating $\kappa_1$ and $\kappa_2$ as regular ElGamal ciphertext components. Additionally, we need to ensure that the biased ciphertext components are not "much" different from a random output of the regular ElGamal encryption scheme, as otherwise the dictator would be able to tell apart regular and anamorphic instances. We now give more details.

The ElGamal 2-message asymmetric anamorphic encryption extension $\Pi_{\mathsf{aEl}} = (\mathsf{aGen}, \mathsf{aEnc}, \mathsf{aDec})$ works as follows: the key generation algorithm $\mathsf{aEl.aGen}$ sets $(\mathsf{apk}, \mathsf{ask}) = (\mathsf{pk}, \mathsf{sk})$ as in $\mathsf{El.Gen}$. In addition, for a uniformly random $\alpha \in \mathbb{Z}_q$, it sets the double keys as $\mathsf{tk} = \alpha$ and $\mathsf{dk} = (g^\alpha, g^{\alpha x})$. On input a regular message vector $\mathbf{msg} = (\mathsf{msg}_1, \mathsf{msg}_2) \in \mathcal{M}^2$, and an anamorphic message $\mathsf{amsg} \in \widehat{\mathcal{M}}$, where $\widehat{\mathcal{M}}$ is a polynomially bounded subspace of $\mathbb{Z}_q$, the encryption algorithm $\mathsf{aEl.aEnc}$ computes $\mathsf{ct}_1 = (r_1, c_1)$ using a uniformly random $\kappa$ on $\mathsf{msg}_1$, identical to $\mathsf{2El.Enc}$. However, for computing the second ciphertext $\mathsf{ct}_2$, instead of choosing a uniformly random element $\kappa_2$ in $\mathbb{Z}_q$, the algorithm implicitly sets $\kappa_2 = \alpha \cdot \kappa + \mathsf{amsg}$ and obtains $r_2 := g^{\alpha\kappa + \mathsf{amsg}}$ and $c_2 := g^{x(\alpha\kappa + \mathsf{amsg})} \cdot \mathsf{msg}_2$. The decryption algorithm $\mathsf{aEl.aDec}$ recovers $\mathsf{amsg}$ by simply returning the discrete logarithm of $r_2 \cdot (r_1)^{-\mathsf{tk}}$, taking advantage of the polynomial message space $\widehat{\mathcal{M}}$. Recall that the dictator has no information on the key pair $(\mathsf{tk}, \mathsf{dk})$, and hence is oblivious to the presence of $\mathsf{amsg}$. We present our construction in Figure 1.

**Correctness, Compatibility and Security Analysis.** We provide proofs of correctness, compatibility, anamorphic security and (s)IND-CPA security of $\Pi_{\mathsf{aEl}}$ below.

**Correctness**: We now show the correctness of $\Pi_{\mathsf{aEl}} = (\mathsf{aGen}, \mathsf{aEnc}, \mathsf{aDec})$ based on the ElGamal PKE scheme. Let $\mathbf{act} = (\mathsf{act}_1, \mathsf{act}_2)$ be the output of $\mathsf{aEl.aEnc}(\mathsf{apk}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg})$, where $\mathsf{act}_1 = (r_1, c_1)$ and $\mathsf{act}_2 = (r_2, c_2)$ are generated using randomness $\kappa$ and $\mathsf{dk}_1^\kappa \cdot g^{\mathsf{amsg}}$. The double encryption key $\mathsf{dk} = (\mathsf{dk}_1, \mathsf{dk}_2) = (g^\alpha, g^{\alpha x})$. Therefore, we have $r_1 = g^\kappa$ and $r_2 = g^{\alpha\kappa + \mathsf{amsg}}$. The anamorphic decryption algorithm knows $\mathsf{tk} = \alpha$, the double decryption key corresponding to $\mathsf{dk}$. Thus, it can compute $g^{\mathsf{amsg}} = r_2 \cdot (r_1)^{-\mathsf{tk}}$. Since $\widehat{\mathcal{M}}$ is polynomially bounded, the decryption algorithm recovers $\mathsf{amsg} \in \widehat{\mathcal{M}}$ by computing the discrete logarithm of $g^{\mathsf{amsg}}$, which is a poly-time algorithm.

[6] Ignoring the final message extraction step natural to most group-based decryption schemes.

$\underline{\mathsf{aGen}(1^\lambda):}$

1. $(\mathsf{sk} = x, \mathsf{pk} = y) \leftarrow \mathsf{El.Gen}(1^\lambda)$
2. $\alpha \leftarrow \mathbb{Z}_q$
3. $\mathsf{ask} \coloneqq \mathsf{sk}, \mathsf{apk} \coloneqq \mathsf{pk}$
4. $\mathsf{dk} \coloneqq (g^\alpha, g^{\alpha x}), \mathsf{tk} \coloneqq \alpha$
5. **return** $(\mathsf{ask}, \mathsf{apk}, \mathsf{dk}, \mathsf{tk})$

$\underline{\mathsf{aDec}(\mathsf{tk}, \mathbf{act}):}$

1. $g^{\mathsf{amsg}} \coloneqq \mathsf{EL.Dec}(\mathsf{tk}, F(\mathbf{act}))$
2. **return** $\log g^{\mathsf{amsg}}$

$\underline{\mathsf{aEnc}(\mathsf{apk}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg}):}$

1. Parse $\mathsf{dk} = (\mathsf{dk}_1, \mathsf{dk}_2)$
2. Parse $\mathbf{msg} = (\mathsf{msg}_1, \mathsf{msg}_2)$
3. $\kappa \leftarrow \mathbb{Z}_q$
4. $\mathsf{act}_1 \coloneqq \mathsf{El.Enc}(\mathsf{apk}, \mathsf{msg}_1; \kappa)$

   // $\mathsf{act}_1 = (r_1, c_1)$

   // $r_1 = g^\kappa, c_1 \coloneqq \mathsf{apk}^\kappa \cdot \mathsf{msg}_1$
5. $\mathsf{act}_2 \coloneqq \mathsf{El.Enc}(\mathsf{apk}, \mathsf{msg}_2; \kappa\alpha + \mathsf{amsg})$

   // $\mathsf{act}_2 = (r_2, c_2)$

   // $r_2 = \mathsf{dk}_1^\kappa \cdot g^{\mathsf{amsg}}$

   // $c_2 = \mathsf{dk}_2^\kappa \cdot \mathsf{apk}^{\mathsf{amsg}} \cdot \mathsf{msg}_2$
6. **return** $\mathbf{act} \coloneqq (\mathsf{act}_1, \mathsf{act}_2)$

$\underline{F(\mathbf{act}):}$

1. Parse $\mathbf{act} = (\mathsf{act}_1, \mathsf{act}_2)$
2. Parse $\mathsf{act}_1 = (r_1, c_1)$, $\mathsf{act}_2 = (r_2, c_2)$
3. **return** $(r_1, r_2)$

Figure 1: 2-AAEE $\Pi_{\mathsf{aEl}}$

**Lemma 2** *ElGamal is AAE-compatible for* $\Pi_{\mathsf{aEl}}$ *described in Figure 1.*

**Proof.** First, we observe that for a given key pair $(\mathsf{pk}, \mathsf{sk})$ of a regular ElGamal PKE scheme, we can set $\mathsf{tk}$ as a uniformly random element $\alpha$ in $\mathbb{Z}_q$, and $\mathsf{dk} = (g^{\mathsf{tk}}, \mathsf{pk}^{\mathsf{tk}})$. Thus, it is easy to see that the double key pair $(\mathsf{dk}, \mathsf{tk})$ can be efficiently computed using only $\mathsf{apk}$. Hence, the AAE satisfies the *on-the-fly key generation* property. Next, to see that $\Pi_{\mathsf{aEl}}$ satisfies the *ciphertext extractibility* property is pretty straightforward. The function $F$ defined in Figure 1 extracts the first components $r_1 = g^\kappa$ and $r_2 = g^{\alpha\kappa + \mathsf{amsg}}$ from $\mathsf{act}_1 = (r_1, c_1), \mathsf{act}_2 = (r_2, c_2)$. Then, applying the regular ElGamal decryption algorithm $\mathsf{EL.Dec}$ with $\mathsf{tk}$ would simply recover $\mathsf{amsg}$ by computing $g^{\mathsf{amsg}} = r_2 \cdot (r_1)^{-\mathsf{tk}}$, which is poly-time as the anamorphic message space is poly-bounded. □

**Theorem 1** *Under the* DDH *assumption for* $\mathbb{G}$*, the ElGamal* 2*-message asymmetric anamorphic encryption extension* $\Pi_{\mathsf{aEl}}$ *described in Figure 1 satisfies anamorphic, fully asymmetric IND-CPA security and sIND-CPA security.*

**Proof.** We prove the theorem in three parts.

**Anamorphic security.** We use a sequence of hybrid games $\mathsf{Game}\ 0, \mathsf{Game}\ (1, j)$ for $j \in [0, Q]$ to prove this theorem. In each hybrid game, we assume $\mathcal{D}$ is a PPT dictator and $Q$ be the total number of queries made by $\mathcal{D}$ to the encryption oracle.

Game 0: This is the anamorphic $\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{aEl}}(\lambda)$. The challenger generates $(\mathsf{ask} = x, \mathsf{apk} = g^x, \mathsf{tk}, \mathsf{dk}) \leftarrow \mathsf{aEl.aGen}(1^\lambda)$, and sends the key pair $(\mathsf{apk}, \mathsf{ask})$ to $\mathcal{D}$. Then, $\mathcal{D}$ makes polynomial number of queries to the encryption oracle with a pair of messages $\{(\mathbf{msg}_i, \mathsf{amsg}_i)\}_{i\in[Q]}$. At the $i$-th query, $\mathcal{D}$ receives a ciphertext $\mathbf{act}_i$ computed as follows:

1. Parse $\mathbf{msg}_i = (\mathsf{msg}_{i_1}, \mathsf{msg}_{i_2})$
2. $\alpha, \kappa_1 \leftarrow \mathbb{Z}_q, \kappa_2 \coloneqq \alpha \cdot \kappa_1 + \mathsf{amsg}_i$.
3. $\mathsf{act}_{i_1} \coloneqq \mathsf{El.Enc}(\mathsf{apk}, \mathsf{msg}_{i_1}; \kappa_1) = (g^{\kappa_1}, g^{x\kappa_1} \cdot \mathsf{msg}_{i_1})$
4. $\mathsf{act}_{i_2} \coloneqq \mathsf{El.Enc}(\mathsf{apk}, \mathsf{msg}_{i_2}; \kappa_2) = (g^{\kappa_2}, g^{x\kappa_2} \cdot \mathsf{msg}_{i_2})$
5. **return** $\mathbf{act}_i \coloneqq (\mathsf{act}_{i_1}, \mathsf{act}_{i_2})$

Game $(1, j)$ **for** $j \in [0, Q]$: We define Game $(1, 0)$ to be identical to Game 0, and Game $(1, j)$ is exactly the same as Game $(1, 0)$ except that the first $j$ (anamorphic) ciphertexts $\{\mathbf{act}_i\}_{i\in[1,j]}$ are computed as follows:

1. Parse $\mathbf{msg}_i = (\mathsf{msg}_{i_1}, \mathsf{msg}_{i_2})$ where $i \le j$
2. $\kappa_1, \kappa_2 \leftarrow \mathbb{Z}_q$
3. $\mathsf{act}_{i_1} \coloneqq \mathsf{El.Enc}(\mathsf{apk}, \mathsf{msg}_{i_1}; \kappa_1) = (g^{\kappa_1}, g^{x\kappa_1} \cdot \mathsf{msg}_{i_1})$
4. $\mathsf{act}_{i_2} \coloneqq \mathsf{El.Enc}(\mathsf{apk}, \mathsf{msg}_{i_2}; \kappa_2) = (g^{\kappa_2}, g^{x\kappa_2} \cdot \mathsf{msg}_{i_2})$
5. **return** $\mathbf{act}_i \coloneqq (\mathsf{act}_{i_1}, \mathsf{act}_{i_2})$

We show that Game $(1, j)$ is indistinguishable from Game $(1, j-1)$ by the DDH assumption. The only difference between Game $(1, j)$ and Game $(1, j-1)$ is in the computation of $\kappa_2$. In particular, $\kappa_2$ is set as $(\alpha \cdot \kappa_1 + \mathsf{amsg}_i)$ in Game $(1, j-1)$, whereas $\kappa_2$ is sampled uniformly random from $\mathbb{Z}_q$ in Game $(1, j)$. Thus the challenger can simulate Game $(1, j)$ with $(g^\alpha, g^{\kappa_1}, g^t)$ when $t = \alpha \cdot \kappa_1$, and Game $(1, j)$, when $t$ is uniformly random in $\mathbb{Z}_q$. Then by the DDH assumption in $\mathbb{G}$, the following holds:

$$\Pr[\alpha, \kappa_1 \leftarrow \mathbb{Z}_q : \mathcal{A}(g^\alpha, g^{\kappa_1}, g^{\alpha\cdot\kappa_1}) = 1] - \Pr[\alpha, \kappa_1, t \leftarrow \mathbb{Z}_q : \mathcal{A}(g^\alpha, g^{\kappa_1}, g^t) = 1]| \le \mathsf{negl}(\lambda)$$

Since $t = \alpha \cdot \kappa_1$ is indistinguishable from uniformly random, $\kappa_2 = t + \mathsf{amsg}_i$ is statistically close to $t$. In other words, the hybrids Game $(1, j)$ and Game $(1, j-1)$ are indistinguishable.

Finally, we observe that Game $(1, Q)$ is exactly the same as $\mathsf{GReal}^{\mathcal{D}}_{\mathsf{2El}}(\lambda)$. This completes the proof of anamorphic security.

**Fully asymmetric IND-CPA security.** We prove the security using hybrid games Game $0, \ldots,$ Game 3. In each hybrid game, we assume $\mathcal{A}$ is a PPT adversary.

Game 0: This is the IND-CPA security experiment $\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda, 0)$ defined as follows:

1: $x, \alpha \leftarrow \mathbb{Z}_q$
2: $\mathsf{ask} \coloneqq x, \mathsf{apk} \coloneqq g^x, \mathsf{dk} = (g^\alpha, g^{\alpha x})$
3: $(\mathbf{msg}_0, \mathbf{msg}_1, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk}, \mathsf{dk})$
4: Parse $\mathbf{msg}_0 = (\mathsf{msg}_{0,1}, \mathsf{msg}_{0,2})$
5: Parse $\mathbf{msg}_1 = (\mathsf{msg}_{1,1}, \mathsf{msg}_{1,2})$
6: Parse $\mathsf{dk} = (\mathsf{dk}_1, \mathsf{dk}_2)$
7: $\kappa_1 \leftarrow \mathbb{Z}_q$
8: Set $\kappa_2 \coloneqq \alpha\kappa_1 + \mathsf{amsg}_0$
9: $\mathsf{act}^*_1 \coloneqq \mathsf{El.Enc}(\mathsf{apk}, \mathsf{msg}_{0,1}; \kappa_1) = (r_1 \coloneqq g^{\kappa_1}, c_1 \coloneqq \mathsf{apk}^{\kappa_1} \cdot \mathsf{msg}_{0,1})$
10: $\mathsf{act}^*_2 \coloneqq \mathsf{El.Enc}(\mathsf{apk}, \mathsf{msg}_{0,2}; \kappa_2)$

$= (r_2 \coloneqq \mathsf{dk}_1^{\kappa_1} \cdot g^{\mathsf{amsg}_0}, c_2 \coloneqq \mathsf{dk}_2^{\kappa_1} \cdot \mathsf{apk}^{\mathsf{amsg}_0} \cdot \mathsf{msg}_{0,2})$

11: **return** $\beta' \leftarrow \mathcal{A}(\mathbf{act}^* = (\mathsf{act}_1^*, \mathsf{act}_2^*))$

Game 1: This is the same as Game 0 except that the challenger samples $\kappa_2$ uniformly at random from $\mathbb{Z}_q$ instead of setting it as $\kappa_2 = \kappa\alpha + \mathsf{amsg}_0$. The indistinguishability follows the DDH assumption. More formally, we design a simulator $\mathcal{B}$ that given a DDH instance $(g^{\kappa_1}, g^{\alpha}, g^{t})$ in the group $\mathbb{G}$, interacts with $\mathcal{A}$ to solve whether $t = \alpha\kappa_1$ or $t$ is uniformly random in $\mathbb{Z}_q$.

Simulator $\mathcal{B}(1^\lambda, g^{\kappa_1}, g^{\alpha}, g^{t})$

1: $x \leftarrow \mathbb{Z}_q$

2: $\mathsf{ask} \coloneqq x, \mathsf{apk} \coloneqq g^x, \mathsf{dk} = (g^\alpha, g^{\alpha x})$

3: $(\mathbf{msg}_0, \mathbf{msg}_1, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk}, \mathsf{dk})$

4: Parse $\mathbf{msg}_0 = (\mathsf{msg}_{0,1}, \mathsf{msg}_{0,2})$

5: Parse $\mathbf{msg}_1 = (\mathsf{msg}_{1,1}, \mathsf{msg}_{1,2})$

6: Parse $\mathsf{dk} = (\mathsf{dk}_1, \mathsf{dk}_2)$

7: $\mathsf{act}_1^* \coloneqq \mathsf{El.Enc}(\mathsf{apk}, \mathsf{msg}_{0,1}; \kappa_1) = (r_1 \coloneqq g^{\kappa_1}, c_1 \coloneqq (g^{\kappa_1})^x \cdot \mathsf{msg}_{0,1})$

8: $\mathsf{act}_2^* \coloneqq \mathsf{El.Enc}(\mathsf{apk}, \mathsf{msg}_{0,2}; t + \mathsf{amsg}_0)$

$= (r_2 \coloneqq g^{t+\mathsf{amsg}_0}, c_2 \coloneqq (g^t)^x \cdot \mathsf{apk}^{\mathsf{amsg}_0} \cdot \mathsf{msg}_{2,0})$

9: **return** $\beta' \leftarrow \mathcal{A}(\mathbf{act}^* = (\mathsf{act}_1^*, \mathsf{act}_2^*))$

If $t = \alpha\kappa_1$, then $\mathcal{B}$ simulates the experiment $\mathsf{GsIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda, 0)$. This is because even though $x$ is chosen randomly during simulation, $(\mathsf{apk}, \mathsf{dk})$ is a valid pair, and hence $\mathbf{act}^*$ is a valid encryption of $\mathsf{amsg}_0$. If $t \leftarrow \mathbb{Z}_q$, then $\kappa_2 = t + \mathsf{amsg}_0$ is uniformly distributed in $\mathbb{G}$, and hence $\mathcal{B}$ simulated Game 1 in $\mathcal{A}$'s view. Thus, Game 1 is indistinguishable from Game 0 by the DDH assumption.

Game 2: This is the same as Game 1 except that the challenger encrypts $\mathbf{msg}_1$ instead of $\mathbf{msg}_0$:

1: $x, \alpha \leftarrow \mathbb{Z}_q$

2: $\mathsf{ask} \coloneqq x, \mathsf{apk} \coloneqq g^x, \mathsf{dk} = (g^\alpha, g^{\alpha x})$

3: $(\mathbf{msg}_0, \mathbf{msg}_1, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk}, \mathsf{dk})$

4: Parse $\mathbf{msg}_0 = (\mathsf{msg}_{0,1}, \mathsf{msg}_{0,2})$

5: Parse $\mathbf{msg}_1 = (\mathsf{msg}_{1,1}, \mathsf{msg}_{1,2})$

6: Parse $\mathsf{dk} = (\mathsf{dk}_1, \mathsf{dk}_2)$

7: $\kappa_1, \kappa_2 \leftarrow \mathbb{Z}_q$

8: $\mathsf{act}_1^* \coloneqq \mathsf{El.Enc}(\mathsf{apk}, \mathsf{msg}_{1,1}; \kappa_1) = (r_1 \coloneqq g^{\kappa_1}, c_1 \coloneqq \mathsf{apk}^{\kappa_1} \cdot \mathsf{msg}_{1,1})$

9: $\mathsf{act}_2^* \coloneqq \mathsf{El.Enc}(\mathsf{apk}, \mathsf{msg}_{1,2}; \kappa_2) = (r_2 \coloneqq g^{\kappa_2}, c_2 \coloneqq \mathsf{apk}^{\kappa_2} \cdot \mathsf{msg}_{1,2})$

10: **return** $\beta' \leftarrow \mathcal{A}(\mathbf{act}^* = (\mathsf{act}_1^*, \mathsf{act}_2^*))$

The indistinguishability between Game 1 and Game 2 follows the underlying ElGamal scheme, in particular from the DDH assumption.

Game 3: This is the same as Game 2 except that the challenger sets $\kappa_2 = \alpha\kappa_1 + \mathsf{amsg}_1$ instead of sampling it uniformly at random from $\mathbb{Z}_q$.

1: $x, \alpha \leftarrow \mathbb{Z}_q$

2: $\mathsf{ask} \coloneqq x, \mathsf{apk} \coloneqq g^x, \mathsf{dk} = (g^\alpha, g^{\alpha x})$

3: $(\mathbf{msg}_0, \mathbf{msg}_1, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk}, \mathsf{dk})$

4: Parse $\mathbf{msg}_0 = (\mathsf{msg}_{0,1}, \mathsf{msg}_{0,2})$

5: Parse $\mathbf{msg}_1 = (\mathsf{msg}_{1,1}, \mathsf{msg}_{1,2})$

6: Parse $\mathsf{dk} = (\mathsf{dk}_1, \mathsf{dk}_2)$

7: $\kappa_1 \leftarrow \mathbb{Z}_q$

8: Set $\kappa_2 \coloneqq \alpha\kappa_1 + \mathsf{amsg}_1$

9: $\mathsf{act}_1^* \coloneqq \mathsf{EI.Enc}(\mathsf{apk}, \mathsf{msg}_{1,1}; \kappa_1) = (r_1 \coloneqq g^{\kappa_1}, c_1 \coloneqq \mathsf{apk}^{\kappa_1} \cdot \mathsf{msg}_{1,1})$

10: $\mathsf{act}_2^* \coloneqq \mathsf{EI.Enc}(\mathsf{apk}, \mathsf{msg}_{1,2}; \kappa_2)$

$= (r_2 \coloneqq \mathsf{dk}_1^{\kappa_1} \cdot g^{\mathsf{amsg}_1}, c_2 \coloneqq \mathsf{dk}_2^{\kappa_1} \cdot \mathsf{apk}^{\mathsf{amsg}_1} \cdot \mathsf{msg}_{1,2})$

11: **return** $\beta' \leftarrow \mathcal{A}(\mathbf{act}^* = (\mathsf{act}_1^*, \mathsf{act}_2^*))$

The indistinguishability follows the DDH assumption as in Game 1.

Finally, we observe that Game 3 is exactly the same as $\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda, 1)$. This concludes the fully asymmetric security of $\Pi_{\mathsf{aEI}}$.

**sIND-CPA security.** We prove the strong IND-CPA security by designing a simulator that plays the role of the challenger in the security game.

We start with a contradiction that there exists a PPT adversary $\mathcal{A}$ that breaks the sIND-CPA security of $\Pi_{\mathsf{aEI}}$ with a non-negligible probability. We then design a simulator $\mathcal{B}$ that given a DDH instance $(g^\kappa, g^\alpha, g^t)$ in the group $\mathbb{G}$, interacts with $\mathcal{A}$ to solve whether $t = \alpha\kappa$ or $t$ is uniformly random in $\mathbb{Z}_q$.

$\mathcal{B}$ prepares the simulation by sampling a uniformly random $x$, and setting $\mathsf{ask} = x$, $\mathsf{apk} = g^x$, $\mathsf{dk} = (g^\alpha, g^{\alpha x})$, and sends $(\mathsf{apk}, \mathsf{ask}, \mathsf{dk})$ to $\mathcal{A}$. When $\mathcal{A}$ returns $(\mathbf{msg}, \mathsf{amsg}_0, \mathsf{amsg}_1)$, $\mathcal{B}$ calculates challenge ciphertext $\mathbf{act}^* = \mathsf{aEI.aEnc}(\mathsf{apk}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg}_\beta)$ for a uniformly random bit $\beta \in \{0, 1\}$, and sends it to $\mathcal{A}$. Finally $\mathcal{A}$ outputs its guess $\beta'$, and $\mathcal{B}$ returns $(\beta == \beta')$. The simulation is formally specified as follows:

Simulator $\mathcal{B}(1^\lambda, g^\kappa, g^\alpha, g^t)$

1: $x \leftarrow \mathbb{Z}_q$

2: $\mathsf{ask} \coloneqq x, \mathsf{apk} \coloneqq g^x, \mathsf{dk} = (g^\alpha, g^{\alpha x})$

3: $(\mathbf{msg}, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk}, \mathsf{ask}, \mathsf{dk})$

4: $\beta \leftarrow \{0, 1\}$

5: Parse $\mathbf{msg} = (\mathsf{msg}_1, \mathsf{msg}_2)$

6: Parse $\mathsf{dk} = (\mathsf{dk}_1, \mathsf{dk}_2)$

7: $\mathsf{act}_1^* \coloneqq \mathsf{EI.Enc}(\mathsf{apk}, \mathsf{msg}_1; \kappa) = (r_1 \coloneqq g^\kappa, c_1 \coloneqq \mathsf{apk}^\kappa \cdot \mathsf{msg}_1)$

$= (r_1 \coloneqq g^\kappa, c_1 \coloneqq (g^\kappa)^x \cdot \mathsf{msg}_1)$

8: $\mathsf{act}_2^* \coloneqq \mathsf{EI.Enc}(\mathsf{apk}, \mathsf{msg}_2; \kappa\alpha + \mathsf{amsg}) = (r_2 \coloneqq \mathsf{dk}_1^\kappa \cdot g^{\mathsf{amsg}_\beta}, c_2 \coloneqq \mathsf{dk}_2^\kappa \cdot \mathsf{apk}^{\mathsf{amsg}_\beta} \cdot \mathsf{msg}_2)$

$= (r_2 \coloneqq g^{t+\mathsf{amsg}_\beta}, c_2 \coloneqq (g^t)^x \cdot \mathsf{apk}^{\mathsf{amsg}_\beta} \cdot \mathsf{msg}_2)$

9: $\beta' \leftarrow \mathcal{A}(\mathbf{act}^* = (\mathsf{act}_1^*, \mathsf{act}_2^*))$

10: **return** $\beta == \beta'$

If $t = \alpha\kappa$, then the distribution of $(\mathsf{apk}, \mathsf{ask})$ and the challenge ciphertext $\mathbf{act}^*$ corresponds exactly to $\mathcal{A}$'s view as in the experiment $\mathsf{GsIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda, \beta)$. This is because even though $x$ is chosen randomly during simulation, $(\mathsf{apk}, \mathsf{ask})$ is a valid pair, and hence $\mathbf{act}^*$ is a valid encryption of $\mathsf{amsg}_\beta$. If $t \leftarrow \mathbb{Z}_q$, then with all but negligible probability $\mathsf{act}^*$ is neither an encryption of $\mathsf{amsg}_0$ nor $\mathsf{amsg}_1$. In fact, $t + \mathsf{amsg}_\beta$ is uniformly distributed in $\mathbb{G}$, and hence $\beta$ is hidden from $\mathcal{A}$'s view. Therefore, if $\mathcal{A}$ breaks the sIND-CPA security of $\Pi_{\mathsf{aEI}}$ with a non-negligible probability then $\mathcal{B}$ can also break the $\mathsf{DDH}$ assumption with a non-negligible probability.

□

## 4.2 Extension of the Dual-Regev Encryption Scheme

We now present an AAE construction based on the Dual-Regev cryptosystem by Gentry, Peikert and Vaikuntanathan [GPV08]. We focus on a simple version that encrypts single bits. The Dual-Regev encryption scheme is a triplet of PPT algorithms $\Pi_{\mathsf{DReg}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ defined over a security parameter $\lambda$ as follows: let $n = n(\lambda)$, $q = q(\lambda)$ and $m = m(\lambda)$ be polynomials in $\lambda$. Let $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ denote the group of reals $[0, 1)$ with modulo 1 addition. Then for $\alpha \in \mathbb{R}^+$, let $\Psi_\alpha$ be the distribution on $\mathbb{T}$ of a normal variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$. Following the notations of [GPV08], we set $m \geq 2n \log q$, $r = \omega(\sqrt{\log m})$, $q \geq 5r(m+1)$, $\alpha \leq 1/(r\sqrt{m+1} \cdot \omega(\sqrt{\log n}))$ and the LWE noise distribution $\chi = \Psi_\alpha$. Dual-Regev with this parameter settings as described above is IND-CPA secure under the $(n, q, \chi)$-$\mathsf{DLWE}$ assumption of dimension $m$.

**Dual-Regev 2-Message AAE.** The Dual-Regev 2-message AAE extension $\Pi_{\mathsf{aDReg}} = (\mathsf{aGen}, \mathsf{aEnc}, \mathsf{aDec})$ works as follows: the key generation $\mathsf{aDReg.aGen}$ sets $(\mathsf{apk}, \mathsf{ask}) = (\mathsf{pk}, \mathsf{sk})$ output by $\mathsf{DReg.Gen}$. It then selects a secret vector $\widehat{\mathbf{k}}$, and sets $\mathsf{tk} \coloneqq \widehat{\mathbf{k}}$, $\mathsf{dk} \coloneqq (\mathbf{A}, \widehat{\mathbf{b}} = \mathbf{A} \cdot \widehat{\mathbf{k}})$. To encrypt a regular message vector $\mathbf{msg} = (\mathsf{msg}_1, \mathsf{msg}_2) \in \mathcal{M}^2$, along with an anamorphic message $\mathsf{amsg} \in \widehat{\mathcal{M}}$, where $\mathcal{M} = \widehat{\mathcal{M}} = \{0, 1\}$, the anamorphic encryption algorithm $\mathsf{aDReg.aEnc}$ samples a uniformly random $\mathbf{s}_1$ in $\mathbb{Z}_q^n$, $\mathbf{e}_1, \mathbf{e}_2$ from $\chi^m$ and $\widehat{y}$ from $\chi$. It then computes $\mathsf{act}_1 = (\mathbf{r}_1^T, c_1)$ as $\mathsf{DReg.Enc}(\mathsf{apk}, \mathsf{msg}_1; \mathbf{s}_1, \mathbf{e}_1^T)$. To encode $\mathsf{amsg}$ in $\mathbf{r}_2^T$ (first component of $\mathsf{act}_2$), it first computes a non-zero solution to $\mathbf{a}_1^T \cdot \mathbf{x} = \mathbf{s}_1^T \cdot \widehat{\mathbf{b}} + \widehat{y} + \mathsf{amsg} \cdot \lfloor \frac{q}{2} \rfloor - e_{2,1} \mod q$, where $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m]$ and $\mathbf{e}_2^T = [e_{2,1}, \ldots, e_{2,m}]$, and sets $\mathbf{s}_2 = \mathbf{x}$. We know that such a solution exists (by the rank-nullity theorem). Then, $\mathsf{act}_2 = (\mathbf{r}_2^T, c_2)$ is computed as $\mathsf{DReg.Enc}(\mathsf{apk}, \mathsf{msg}_2; \mathbf{s}_2, \mathbf{e}_2^T)$. Finally, the anamorphic ciphertext $\mathbf{act}$ is $(\mathsf{act}_1, \mathsf{act}_2)$. Let $\mathsf{proj} : \mathbb{Z}_q^m \to \mathbb{Z}_q$ be a projection onto $\mathbb{Z}_q$. To recover $\mathsf{amsg}$, the anamorphic decryption algorithm $\mathsf{aDReg.aDec}$ extracts the first element of $\mathbf{r}_2^T$ as $\widehat{c} = \mathsf{proj}_1(\mathbf{r}_2^T)$, followed by computing $\mathsf{DReg.Dec}$ on $\widehat{\mathbf{r}}^T = \mathbf{r}_1^T$ and $\widehat{c}$, using $\mathsf{tk}$. We note that $\mathsf{amsg}$ is hidden from the dictator's view as it has no information on the double keys. See Figure 2 for our construction.

**Correctness, Compatibility and Security Analysis.** We provide proofs of correctness, compatibility, anamorphic security and (s)IND-CPA security of $\Pi_{\mathsf{aDReg}}$ below.

**Correctness**: We now show the correctness of the $\Pi_{\mathsf{aDReg}} = (\mathsf{aGen}, \mathsf{aEnc}, \mathsf{aDec})$ based on the Dual-Regev PKE scheme. Let the encryption double key be $\mathsf{dk} = (\mathsf{dk}_1, \mathsf{dk}_2) = (\mathbf{A}, \widehat{\mathbf{b}} = \mathbf{A} \cdot \widehat{\mathbf{k}})$. Let $\mathbf{act} = (\mathsf{act}_1, \mathsf{act}_2)$ output by $\mathsf{aDReg.aEnc}(\mathsf{apk}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg})$, where $\mathsf{act}_1 = (\mathbf{r}_1^T, c_1)$ and $\mathsf{act}_2 = (\mathbf{r}_2^T, c_2)$ generated using randomness $\mathbf{s}_1$ and $\mathbf{s}_2$ such that $\mathbf{r}_1^T = \mathbf{s}_1^T \cdot \mathsf{dk}_1 + \mathbf{e}_1^T$ and $\mathbf{r}_2^T = \mathbf{s}_2^T \cdot \mathsf{dk}_1 + \mathbf{e}_2^T$, where $\mathbf{s}_2$ is set as a solution to $\mathbf{a}_1^T \cdot \mathbf{x} = \mathbf{s}_1^T \cdot \mathsf{dk}_2 + \widehat{y} + \mathsf{amsg} \cdot \lfloor \frac{q}{2} \rfloor - e_{2,1} \mod q$, where $\mathsf{dk}_1 = [\mathbf{a}_1, \ldots, \mathbf{a}_m] \in \mathbb{Z}_q^{n \times m}$, $\mathbf{e}_2^T = [e_{2,1}, \ldots, e_{2,m}] \in \chi^{1 \times m}$ and $\widehat{y} \in \chi$. Then, observe that the first element of the vector $\mathbf{r}_2^T$ is $\mathbf{s}_1^T \cdot \mathsf{dk}_2 + \widehat{y} + \mathsf{amsg} \cdot \lfloor \frac{q}{2} \rfloor \mod q$. The anamorphic decryption algorithm knows $\mathsf{tk} = \widehat{\mathbf{k}}$, the decryption double key corresponding to $\mathsf{dk}$. It calls the function $F(\mathbf{act})$, which extracts the first element of $\mathbf{r}_2^T$ using a projection function and assigns the value to $\widehat{c}$. Thus it can compute $\mathsf{amsg}$ from $\widehat{\mathbf{r}}^T = \mathbf{r}_1^T = \mathbf{s}_1^T \cdot \mathsf{dk}_1 + \mathbf{e}_1^T$ and $\widehat{c} = \mathbf{s}_1^T \cdot \mathsf{dk}_2 + \widehat{y} + \mathsf{amsg} \cdot \lfloor \frac{q}{2} \rfloor \mod q$, by returning $\mathsf{amsg} = 0$ if $|\widehat{c} - \widehat{\mathbf{r}}^T \cdot \mathsf{tk}| \leq \frac{q}{4}$, and $\mathsf{amsg} = 1$ otherwise. Note that, $|\widehat{c} - \widehat{\mathbf{r}}^T \cdot \mathsf{tk}|$ simply reduces to $\mathsf{amsg} \cdot \lfloor \frac{q}{2} \rfloor$ plus an LWE noise $\widehat{y} - \mathbf{e}_2^T \cdot \widehat{\mathbf{k}} \mod q$, whose norm is smaller than $\frac{q}{5}$, except with a negligible probability in $\lambda$.

$\underline{\mathsf{aGen}(1^\lambda):}$

1. $(\mathsf{sk} := \mathbf{k}, \mathsf{pk} := (\mathbf{A}, \mathbf{b} = \mathbf{A}\cdot\mathbf{k})) \leftarrow \mathsf{DReg.Gen}(1^\lambda)$
2. $\widehat{\mathbf{k}} \leftarrow \{0,1\}^m$
3. $\mathsf{ask} := \mathsf{sk}$, $\mathsf{apk} := \mathsf{pk}$
4. $\mathsf{tk} := \widehat{\mathbf{k}}$, $\mathsf{dk} := (\mathbf{A}, \widehat{\mathbf{b}} = \mathbf{A}\cdot\widehat{\mathbf{k}})$
5. **return** $(\mathsf{ask}, \mathsf{apk}, \mathsf{tk}, \mathsf{dk})$

$\underline{\mathsf{aDec}(\mathsf{tk}, \mathbf{act}):}$

1. $\mathsf{amsg} := \mathsf{DReg.Dec}(\mathsf{tk}, F(\mathbf{act}))$
2. **return** $\mathsf{amsg}$

$\underline{F(\mathbf{act}):}$

1. Parse $\mathbf{act} = (\mathsf{act}_1, \mathsf{act}_2)$
2. Parse $\mathsf{act}_1 = (\mathbf{r}_1^T, c_1)$,
   $\mathsf{act}_2 = (\mathbf{r}_2^T, c_2)$
3. $\mathbf{r}_2^T = [d_1, \ldots, d_m]$
4. $\widehat{c} := \mathsf{proj}_1(\mathbf{r}_2^T)$, $\widehat{\mathbf{r}}^T := \mathbf{r}_1^T$
5. **return** $(\widehat{\mathbf{r}}^T, \widehat{c})$

$\underline{\mathsf{aEnc}(\mathsf{apk}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg}):}$

1. Parse $\mathbf{msg} = (\mathsf{msg}_1, \mathsf{msg}_2)$
2. $\mathbf{s}_1 \leftarrow \mathbb{Z}_q^n$, $\mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi^m$, $\widehat{y} \leftarrow \chi$
3. $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m]$, $\mathbf{e}_2^T = [e_{2,1}, \ldots, e_{2,m}]$
   $\mathbf{a}_1^T = [a_{1,1}, \ldots, a_{1,n}]$
4. **for** $i \in \{1, \ldots, n-1\}$: $x_i \leftarrow \mathbb{Z}_q$
5. $x_n = \Big( -\sum_{i=1}^{n-1} x_i a_{1,i} + \mathbf{s}_1^T \cdot \widehat{\mathbf{b}} + \widehat{y} + \mathsf{amsg} \cdot \lfloor \frac{q}{2} \rfloor - e_{2,1} \Big) \cdot a_{1,n}^{-1} \mod q$
6. $\mathbf{x}^T = [x_1, \ldots, x_n]$
7. $\mathbf{s}_2 := \mathbf{x}$
8. $\mathsf{act}_1 := \mathsf{DReg.Enc}(\mathsf{apk}, \mathsf{msg}_1; \mathbf{s}_1, \mathbf{e}_1^T)$
   //$\mathsf{act}_1 = (\mathbf{r}_1^T, c_1)$
   // $\mathbf{r}_1^T = \mathbf{s}_1^T \cdot \mathbf{A} + \mathbf{e}_1^T$
   //$c_1 = \mathbf{s}_1^T \cdot \mathbf{b} + y_1 + \mathsf{msg}_1 \cdot \lfloor \frac{q}{2} \rfloor \mod q$
9. $\mathsf{act}_2 := \mathsf{DReg.Enc}(\mathsf{apk}, \mathsf{msg}_2; \mathbf{s}_2, \mathbf{e}_2^T))$
   // $\mathsf{act}_2 = (\mathbf{r}_2^T, c_2)$
   // $\mathbf{r}_2^T = \mathbf{s}_2^T \cdot \mathbf{A} + \mathbf{e}_2^T$
   // $c_2 = \mathbf{s}_2^T \cdot \mathbf{b} + y_2 + \mathsf{msg}_2 \cdot \lfloor \frac{q}{2} \rfloor \mod q$
10. **return act** $:= (\mathsf{act}_1, \mathsf{act}_2)$

Figure 2: 2-AAEE $\Pi_{\mathsf{aDReg}}$

**Lemma 3** *Dual-Regev is AAE-compatible for* $\Pi_{\mathsf{aDReg}}$ *described in Figure 2.*

**Proof.** First, observe that $\mathsf{aDReg.aGen}(1^\lambda)$ outputs $\mathsf{ask} = \mathbf{k}$, $\mathsf{apk} = (\mathbf{A}, \mathbf{b} = \mathbf{A}\cdot\mathbf{k})$, $\mathsf{tk} = \widehat{\mathbf{k}}$ and $\mathsf{dk} = (\mathbf{A}, \widehat{\mathbf{b}} = \mathbf{A}\cdot\widehat{\mathbf{k}})$, where $\mathbf{k}$ and $\widehat{\mathbf{k}}$ are sampled uniformly at random from $\{0,1\}^m$. Therefore, the key pair $(\mathsf{apk}, \mathsf{ask})$ has identical distributions with $\{(\mathsf{pk}, \mathsf{sk}) : \mathsf{sk} = \widetilde{\mathbf{k}}, \mathsf{pk} = (\mathbf{A}, \widetilde{\mathbf{b}} = \mathbf{A}\cdot\widetilde{\mathbf{k}})$ s.t. $\widetilde{\mathbf{k}} \leftarrow \{0,1\}^m\}$, which is essentially the distribution of secret and public keys of a Dual-Regev PKE scheme. Additionally, it is easy to see that the double key pair $(\mathsf{dk}, \mathsf{tk})$ can be efficiently computed using only $\mathsf{apk}$. Hence, the AAE satisfies *on-the-fly double key generation* property. Next, observe that the function $F$ defined in Figure 2 takes as input $\mathbf{act} \leftarrow \mathsf{aDReg.aEnc}(\mathsf{apk}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg})$, and parses it as $\mathsf{act}_1 = (\mathbf{r}_1^T, c_1)$ and $\mathsf{act}_2 = (\mathbf{r}_2^T, c_2)$. It then extracts $\mathbf{r}_1^T$ (call it $\widehat{\mathbf{r}}^T$) and the first component of $\mathbf{r}_2^T$ (call it $\widehat{c}$). Note that $(\widehat{\mathbf{r}}^T, \widehat{c})$ is a valid Dual-Regev ciphertext as $\widehat{\mathbf{r}}^T = \mathbf{s}_1^T \cdot \mathbf{A} + \mathbf{e}_1^T$ and $\widehat{c} = \mathbf{s}_1^T \cdot \widehat{\mathbf{b}} + \widehat{y} + \mathsf{amsg} \cdot \lfloor \frac{q}{2} \rfloor \mod q$. Thus, $\mathsf{amsg}$ can be extracted by executing $\mathsf{DReg.Dec}$ using $\mathsf{tk} = \widehat{\mathbf{k}}$, which implies that the AAE satisfies the *ciphertext extractibility* property.

Hence we conclude that the Dual-Regev PKE is AAE-compatible. □

**Theorem 2** *Let us consider* $m \geq 2n\log q$, $r = \omega(\sqrt{\log m})$, $q \geq 5r(m+1)$, $\alpha \leq 1/(r\sqrt{m+1} \cdot \omega(\sqrt{\log n}))$ *and* $\chi = \Psi_\alpha$. *Then, under the* $(n, q, \chi)$-$\mathsf{DLWE}$ *assumption of dimension* $m$, *the Dual-Regev* 2*-message anamorphic encryption extension* $\Pi_{\mathsf{aDReg}}$ *described in Fig. 2 satisfies anamorphic, fully asymmetric IND-CPA security and sIND-CPA security.*

**Proof.** We prove the theorem in three parts.

**Anamorphic security.** We use a sequence of hybrid games $\mathsf{Game}\ 0, \mathsf{Game}\ (1,j)$ for $j \in [0,Q]$ to prove this theorem. In each hybrid game, we assume $\mathcal{D}$ is a PPT dictator and $Q$ be the total number of queries made by the dictator to the encryption oracle.

Game 0**:** This is the anamorphic $\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{aDReg}}(\lambda)$. The challenger generates $(\mathsf{apk} = (\mathbf{A}, \mathbf{b} = \mathbf{A}\cdot\mathbf{k}), \mathsf{ask} = \mathbf{k}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aDReg.aGen}(1^\lambda)$, and sends the key pair $(\mathsf{apk}, \mathsf{ask})$ to $\mathcal{D}$. Then $\mathcal{D}$ makes polynomial number of queries to the encryption oracle with a pair of messages $\{(\mathbf{msg}_i, \mathsf{amsg}_i)\}_{i\in[Q]}$. At the $i$-th query, the dictator receives a ciphertext $\mathbf{act}_i$ computed as follows:

1. Parse $\mathbf{msg}_i = (\mathsf{msg}_{i_1}, \mathsf{msg}_{i_2})$
2. $\widehat{\mathbf{k}} \leftarrow \{0,1\}^m$, $\mathbf{s}_1 \leftarrow \mathbb{Z}_q^n, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi^m, y_1, y_2, \widehat{y} \leftarrow \chi$
3. $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m] \in \mathbb{Z}_q^{n\times m}, \mathbf{e}_2^T = [e_{2,1}, \ldots, e_{2,m}] \in \chi^{1\times m}$
4. $\widehat{\mathbf{b}} \coloneqq \mathbf{A}\cdot\widehat{\mathbf{k}}$
5. **for** $i \in \{1, \ldots, n-1\}$, $x_i \leftarrow \mathbb{Z}_q$
6. **end for**
7. $x_n = (-\sum_{i=1}^{n-1} x_i a_{1,i} + \mathbf{s}_1^T \cdot \widehat{\mathbf{b}} + \widehat{y} + \mathsf{amsg}_i \cdot \lfloor \frac{q}{2} \rfloor - e_{2,1}) \cdot a_{1,n}^{-1} \mod q$
8. $\mathbf{x}^T = [x_1, \ldots, x_n]$
9. $\mathbf{s}_2 \coloneqq \mathbf{x}$
10. $\mathsf{act}_{i_1} \coloneqq \mathsf{DReg.Enc}(\mathsf{apk}, \mathsf{msg}_{i_1}; \mathbf{s}_1, \mathbf{e}_1^T)$
$= (\mathbf{s}_1^T \cdot \mathbf{A} + \mathbf{e}_1^T, \mathbf{s}_1^T \cdot \mathbf{b} + y_1 + \mathsf{msg}_{i_1} \cdot \lfloor \frac{q}{2} \rfloor \mod q)$
11. $\mathsf{act}_{i_2} \coloneqq \mathsf{DReg.Enc}(\mathsf{apk}, \mathsf{msg}_{i_2}; \mathbf{s}_2, \mathbf{e}_2^T)$
$= (\mathbf{s}_2^T \cdot \mathbf{A} + \mathbf{e}_2^T, \mathbf{s}_2^T \cdot \mathbf{b} + y_2 + \mathsf{msg}_{i_2} \cdot \lfloor \frac{q}{2} \rfloor \mod q)$
12. **return** $\mathbf{act}_i \coloneqq (\mathsf{act}_{i_1}, \mathsf{act}_{i_2})$

$\mathsf{Game}\ (1,j)$ **for** $j \in [0,Q]$**:** We define $\mathsf{Game}\ (1,0)$ to be identical to $\mathsf{Game}\ 0$ and $\mathsf{Game}\ (1,j)$ is exactly the same as $\mathsf{Game}\ (1,0)$ except that the first $j$ (anamorphic) ciphertexts $\{\mathbf{act}_i\}_{i\in[1,j]}$ are computed as follows:

1. Parse $\mathbf{msg}_i = (\mathsf{msg}_{i_1}, \mathsf{msg}_{i_2})$ where $i \leq j$
2. $\mathbf{s}_1, \mathbf{s}_2 \leftarrow \mathbb{Z}_q^n, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi^m, y_1, y_2 \leftarrow \chi$
3. $\mathsf{msg}_{i_1} \leftarrow \{0,1\}$, $\mathsf{msg}_{i_2} \coloneqq \mathsf{msg}_i \oplus \mathsf{msg}_{i_1}$
4. $\mathsf{act}_{i_1} \coloneqq \mathsf{DReg.Enc}(\mathsf{apk}, \mathsf{msg}_{i_1}; \mathbf{s}_1, \mathbf{e}_1^T)$
$= (\mathbf{s}_1^T \cdot \mathbf{A} + \mathbf{e}_1^T, \mathbf{s}_1^T \cdot \mathbf{b} + y_1 + \mathsf{msg}_{i_1} \cdot \lfloor \frac{q}{2} \rfloor \mod q)$
5. $\mathsf{act}_{i_2} \coloneqq \mathsf{DReg.Enc}(\mathsf{apk}, \mathsf{msg}_{i_2}; \mathbf{s}_2, \mathbf{e}_2^T)$
$= (\mathbf{s}_2^T \cdot \mathbf{A} + \mathbf{e}_2^T, \mathbf{s}_2^T \cdot \mathbf{b} + y_2 + \mathsf{msg}_{i_2} \cdot \lfloor \frac{q}{2} \rfloor \mod q)$
6. **return** $\mathbf{act}_i \coloneqq (\mathsf{act}_{i_1}, \mathsf{act}_{i_2})$

We show that $\mathsf{Game}\ (1,j)$ is indistinguishable from $\mathsf{Game}\ (1,j-1)$ by the $\mathsf{DLWE}$ assumption. The only difference between $\mathsf{Game}\ (1,j)$ and $\mathsf{Game}\ (1,j-1)$ is in the computation of the vector $\mathbf{s}_2$. In $\mathsf{Game}\ (1,j)$, $\mathbf{s}_2$ is sampled uniformly at random from $\mathbb{Z}_q^n$, whereas in $\mathsf{Game}\ (1,j-1)$, $\mathbf{s}_2$ is set as $\mathbf{x}$ where $\mathbf{x}^T = [x_1,\ldots,x_n]$ such that the first $(n-1)$ elements of the sequence are uniformly random values in $\mathbb{Z}_q$, and $x_n = (-\sum_{i=1}^{n-1} x_i a_{1,i} + \mathbf{s}_1^T \cdot \widehat{\mathbf{b}} + \widehat{y} + \mathsf{amsg}_i \cdot \lfloor \frac{q}{2} \rfloor - e_{2,1}) \cdot a_{1,n}^{-1} \mod q$, where the $a_i$'s are uniformly random elements of $\mathbb{Z}_q^m$. First, we observe that by the leftover hash lemma (see Lemma 1) $(\mathbf{A}, \mathbf{A}\widehat{\mathbf{k}} = \widehat{\mathbf{b}})$ is indistinguishable from $(\mathbf{A}, \widehat{\mathbf{b}} \leftarrow \mathbb{Z}_q^n)$ in $\mathcal{D}$'s view (since $m = \Omega(n \log q)$ and $\widehat{\mathbf{k}}$ is not given to the dictator). Next, by the $\mathsf{DLWE}$ assumption, we have

$$
\begin{aligned}
&((\mathbf{A}||\widehat{\mathbf{b}}),\ \mathbf{s}_1^T \cdot (\mathbf{A}||\widehat{\mathbf{b}}) + (\mathbf{e}_1^T, \widehat{y})) \\
&\approx_c ((\mathbf{A}||\widehat{\mathbf{b}}),\ (\mathbf{t} \leftarrow \mathbb{Z}_q^m, u \leftarrow \mathbb{Z}_q) \mod q \qquad \text{(by } \mathsf{DLWE}\text{)} \\
&\approx_c ((\mathbf{A}||\widehat{\mathbf{b}}),\ (\mathbf{s}_1^T\mathbf{A} + \mathbf{e}_1^T, u \leftarrow \mathbb{Z}_q) \mod q \qquad \text{(by } \mathsf{DLWE}\text{)}.
\end{aligned}
$$

Observe that we can use $\mathbf{s}_1^T\mathbf{A} + \mathbf{e}_1^T$ and $\mathbf{k}$ to simulate $\mathsf{act}_{i_1}$ (by implicitly setting $y_1 = \mathbf{e}_1^T \cdot \mathbf{k} + y_1' \in \chi$ for a uniformly sampled $y_1' \leftarrow \chi$) and replace $(\mathbf{s}_1^T \cdot \widehat{\mathbf{b}} + \widehat{y})$ with an uniformly random value $u$. Thus, $x_n$ appears uniformly random in $\mathbb{Z}_q$, which essentially hides $\mathsf{amsg}_\beta$ within $\mathsf{act}_j$. In other words, the hybrids $\mathsf{Game}\ (1,j)$ and $\mathsf{Game}\ (1,j-1)$ are indistinguishable.
Finally, we observe that $\mathsf{Game}\ (1,Q)$ is exactly the same as $\mathsf{GReal}^{\mathcal{D}}_{\mathsf{2DReg}}(\lambda)$. This completes the proof.

**Fully asymmetric IND-CPA security.** We prove the security using hybrid games $\mathsf{Game}\ 0,\ldots,\mathsf{Game}\ 3$. In each hybrid game, we assume $\mathcal{A}$ is a PPT adversary.

<u>Game 0:</u> This is the IND-CPA security experiment $\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AAE}}(\lambda, 0)$ defined as follows:

1: $\mathbf{k}, \widehat{\mathbf{k}} \leftarrow \{0,1\}^m$
2: $\mathsf{ask} \coloneqq \mathbf{k}, \mathsf{apk} \coloneqq (\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{k}), \mathsf{dk} = (\mathbf{A}, \widehat{\mathbf{b}} = \mathbf{A}\widehat{\mathbf{k}})$
3: $(\mathbf{msg}_0, \mathbf{msg}_1, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk}, \mathsf{dk})$
4: Parse $\mathbf{msg}_0 = (\mathsf{msg}_{0,1}, \mathsf{msg}_{0,2})$
5: Parse $\mathbf{msg}_1 = (\mathsf{msg}_{1,1}, \mathsf{msg}_{1,2})$
6: Parse $\mathsf{dk} = (\mathsf{dk}_1, \mathsf{dk}_2)$
7: $\mathbf{s}_1 \leftarrow \mathbb{Z}_q^n, \mathbf{e}_2 \leftarrow \chi^m, \widehat{y}, y_1, y_2 \leftarrow \chi$
8: $\mathsf{dk}_1 = [\mathbf{a}_1,\ldots,\mathbf{a}_m] \in \mathbb{Z}_q^{n\times m}, \mathbf{e}_2^T = [e_{2,1},\ldots,e_{2,m}] \in \chi^{1\times m}$
9: **for** $i \in \{1,\ldots,n-1\}$, $x_i \leftarrow \mathbb{Z}_q$
10: **end for**
11: $x_n = (-\sum_{i=1}^{n-1} x_i a_{1,i} + \mathbf{s}_1^T \cdot \mathsf{dk}_2 + \widehat{y} + \mathsf{amsg}_0 \cdot \lfloor \frac{q}{2} \rfloor - e_{2,1}) \cdot a_{1,n}^{-1} \mod q$
12: $\mathbf{x}^T = [x_1,\ldots,x_n]$
13: $\mathbf{s}_2 \coloneqq \mathbf{x}$
14: $\mathsf{act}_1^* \coloneqq \mathsf{DReg.Enc}(\mathsf{apk}, \mathsf{msg}_{0,1}; \mathbf{s}_1, \mathbf{e}_1^T, y_1) = (\mathbf{r}_1^T, c_1)$
// $\mathbf{r}_1^T \coloneqq \mathbf{s}_1^T \cdot \mathsf{dk}_1 + \mathbf{e}_1^T$, $c_1 \coloneqq \mathbf{s}_1^T \cdot \mathbf{b} + y_1 + \mathsf{msg}_{0,1} \cdot \lfloor \frac{q}{2} \rfloor \mod q$
15: $\mathsf{act}_2^* \coloneqq \mathsf{DReg.Enc}(\mathsf{pk}, \mathsf{msg}_{0,2}; \mathbf{s}_2, \mathbf{e}_2^T, y_2) = (\mathbf{r}_2^T, c_2)$
// $\mathbf{r}_2^T \coloneqq \mathbf{s}_2^T \cdot \mathsf{dk}_1 + \mathbf{e}_2^T$, $c_2 \coloneqq \mathbf{s}_2^T \cdot \mathbf{b} + y_2 + \mathsf{msg}_{0,2} \cdot \lfloor \frac{q}{2} \rfloor \mod q$
16: **return** $\beta' \leftarrow \mathcal{A}(\mathbf{act}^* = (\mathsf{act}_1^*, \mathsf{act}_2^*))$

Game 1: This is the same as Game 0 except that the challenger samples $x_n$ uniformly at random from $\mathbb{Z}_q$ instead of setting it as $(\sum_{i=1}^{n-1} x_i a_{1,i} + \mathbf{s}_1^T \cdot \mathsf{dk}_2 + \widehat{y} + \mathsf{amsg}_0 \cdot \lfloor \frac{q}{2} \rfloor - e_{2,1}) \cdot a_{1,n}^{-1} \mod q$. The indistinguishability follows the leftover hash lemma and DLWE assumption. First, we observe that by the leftover hash lemma $(\mathbf{A}, \mathbf{A}\widehat{\mathbf{k}} = \widehat{\mathbf{b}})$ is indistinguishable from $(\mathbf{A}, \widehat{\mathbf{b}} \leftarrow \mathbb{Z}_q^n)$ in the $\mathcal{A}$'s view (since $m = \Omega(n \log q)$). More formally, we design a simulator $\mathcal{B}$ that breaks the following indistinguishability supported by the DLWE assumption,

$$((\mathbf{A}||\widehat{\mathbf{b}}),\ \mathbf{s}_1^T \cdot (\mathbf{A}||\widehat{\mathbf{b}}) + (\mathbf{e}_1^T, \widehat{y})) \approx_c ((\mathbf{A}||\widehat{\mathbf{b}}),\ (\mathbf{s}_1^T \mathbf{A} + \mathbf{e}_1^T, u \leftarrow \mathbb{Z}_q) \mod q\ .$$

Simulator $\mathcal{B}(1^\lambda, \mathbf{A}, \widehat{\mathbf{b}}, \mathbf{s}_1^T \mathbf{A} + \mathbf{e}_1^T, u)$

1: $\mathbf{k} \leftarrow \{0,1\}^m$

2: $\mathsf{ask} \coloneqq \mathbf{k}, \mathsf{apk} \coloneqq (\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{k}), \mathsf{dk} = (\mathbf{A}, \widehat{\mathbf{b}})$

3: $(\mathbf{msg}_0, \mathbf{msg}_1, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk}, \mathsf{dk})$

4: Parse $\mathbf{msg}_0 = (\mathsf{msg}_{0,1}, \mathsf{msg}_{0,2})$

5: Parse $\mathbf{msg}_1 = (\mathsf{msg}_{1,1}, \mathsf{msg}_{1,2})$

6: Parse $\mathsf{dk} = (\mathsf{dk}_1, \mathsf{dk}_2)$

7: $\mathbf{e}_2 \leftarrow \chi^m, y_1', y_2 \leftarrow \chi$

8: $\mathsf{dk}_1 = [\mathbf{a}_1, \ldots, \mathbf{a}_m] \in \mathbb{Z}_q^{n \times m}, \mathbf{e}_2^T = [e_{2,1}, \ldots, e_{2,m}] \in \chi^{1 \times m}$

9: **for** $i \in \{1, \ldots, n-1\}$, $x_i \leftarrow \mathbb{Z}_q$

10: **end for**

11: $x_n = (-\sum_{i=1}^{n-1} x_i a_{1,i} + u + \mathsf{amsg}_0 \cdot \lfloor \frac{q}{2} \rfloor - e_{2,1}) \cdot a_{1,n}^{-1} \mod q$

12: $\mathbf{x}^T = [x_1, \ldots, x_n]$

13: $\mathbf{s}_2 \coloneqq \mathbf{x}$

14: $\mathsf{act}_1^* \coloneqq \mathsf{DReg.Enc}(\mathsf{apk}, \mathsf{msg}_{0,1}; \mathbf{s}_1, \mathbf{e}_1^T, y_1 = \mathbf{e}_1^T \cdot \mathbf{k} + y_1') = (\mathbf{r}_1^T, c_1)$

// $\mathbf{r}_1^T \coloneqq \mathbf{s}_1^T \cdot \mathbf{A} + \mathbf{e}_1^T$, $c_1 \coloneqq (\mathbf{s}_1^T \mathbf{A} + \mathbf{e}_1^T)\mathbf{k} + \mathsf{msg}_{0,1} \cdot \lfloor \frac{q}{2} \rfloor \mod q$

15: $\mathsf{act}_2^* \coloneqq \mathsf{DReg.Enc}(\mathsf{pk}, \mathsf{msg}_{0,2}; \mathbf{s}_2, \mathbf{e}_2^T, y_2) = (\mathbf{r}_2^T, c_2)$

// $\mathbf{r}_2^T \coloneqq \mathbf{s}_2^T \cdot \mathsf{dk}_1 + \mathbf{e}_2^T$, $c_2 \coloneqq \mathbf{s}_2^T \cdot \mathbf{b} + y_2 + \mathsf{msg}_{0,2} \cdot \lfloor \frac{q}{2} \rfloor \mod q$

16: **return** $\beta' \leftarrow \mathcal{A}(\mathbf{act}^* = (\mathsf{act}_1^*, \mathsf{act}_2^*))$

When $u = \mathbf{s}_1^T \cdot \widehat{\mathbf{b}} + \widehat{y}$, then the distribution of $(\mathsf{apk}, \mathsf{dk})$ and the challenge ciphertext $\mathbf{act}^*$ corresponds to $\mathcal{A}$'s view are the same as in the experiment $\mathsf{GIND\text{-}CPA}_{\mathsf{AAE}}^{\mathcal{A}}(\lambda, 0)$. This follows from the anamorphic key-pair being perfectly valid, which makes $\mathsf{act}_1^*$ a valid encryption of $\mathsf{msg}_{0,1}$ under $\mathsf{apk} = (\mathbf{A}, \mathbf{b})$ with $y_1 = \mathbf{e}_1^T \cdot \mathbf{k} + y_1'$. Moreover, $x_n$ is distributed as in the original experiment. On the other hand, if $u$ is uniformly random in $\mathbb{Z}_q$, then (in line 11) $u + \mathsf{amsg}_0 \cdot \lfloor \frac{q}{2} \rfloor$ becomes uniformly random over $\mathbb{Z}_q$ (under the DLWE assumption). In other words, $x_n$ is uniformly distributed over $\mathbb{Z}_q$ and $\mathcal{B}$ simulates Game 1. Thus, Game 1 is indistinguishable from Game 0 by the leftover hash lemma and the DLWE assumption.

Game 2: This is the same as Game 1 except that the challenger encrypts $\mathbf{msg}_1$ instead of $\mathbf{msg}_0$:

1: $\mathbf{k}, \widehat{\mathbf{k}} \leftarrow \{0,1\}^m$

2: $\mathsf{ask} \coloneqq \mathbf{k}, \mathsf{apk} \coloneqq (\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{k}), \mathsf{dk} = (\mathbf{A}, \widehat{\mathbf{b}} = \mathbf{A}\widehat{\mathbf{k}})$

3: $(\mathbf{msg}_0, \mathbf{msg}_1, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk}, \mathsf{dk})$

4: Parse $\mathbf{msg}_0 = (\mathsf{msg}_{0,1}, \mathsf{msg}_{0,2})$

5: Parse $\mathbf{msg}_1 = (\mathsf{msg}_{1,1}, \mathsf{msg}_{1,2})$

6: Parse $\mathsf{dk} = (\mathsf{dk}_1, \mathsf{dk}_2)$

7: $\mathbf{s}_1, \mathbf{s}_2 \leftarrow \mathbb{Z}_q^n, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi^m, \widehat{y}, y_1, y_2 \leftarrow \chi$,

8: $\mathsf{act}_1^* \coloneqq \mathsf{DReg.Enc}(\mathsf{apk}, \mathsf{msg}_{1,1}; \mathbf{s}_1, \mathbf{e}_1^T, y_1) = (\mathbf{r}_1^T, c_1)$

// $\mathbf{r}_1^T \coloneqq \mathbf{s}_1^T \cdot \mathsf{dk}_1 + \mathbf{e}_1^T$, $c_1 \coloneqq \mathbf{s}_1^T \cdot \mathbf{b} + y_1 + \mathsf{msg}_{1,1} \cdot \lfloor \frac{q}{2} \rfloor \mod q$

9: $\mathsf{act}_2^* \coloneqq \mathsf{DReg.Enc}(\mathsf{pk}, \mathsf{msg}_{1,2}; \mathbf{s}_2, \mathbf{e}_2^T, y_2) = (\mathbf{r}_2^T, c_2)$

// $\mathbf{r}_2^T \coloneqq \mathbf{s}_2^T \cdot \mathsf{dk}_1 + \mathbf{e}_2^T$, $c_2 \coloneqq \mathbf{s}_2^T \cdot \mathbf{b} + y_2 + \mathsf{msg}_{1,2} \cdot \lfloor \frac{q}{2} \rfloor \mod q$

10: **return** $\beta' \leftarrow \mathcal{A}(\mathbf{act}^* = (\mathsf{act}_1^*, \mathsf{act}_2^*))$

The indistinguishability between Game 1 and Game 2 follows the underlying Dual-Regev encryption scheme, in particular from the leftover hash lemma and DLWE assumption.

Game 3**:** This is the same as Game 2 except that the challenger sets $x_n = (-\sum_{i=1}^{n-1} x_i a_{1,i} + \mathbf{s}_1^T \cdot \mathsf{dk}_2 + \widehat{y} + \mathsf{amsg}_1 \cdot \lfloor \frac{q}{2} \rfloor - e_{2,1}) \cdot a_{1,n}^{-1} \mod q$ instead of sampling it uniformly at random from $\mathbb{Z}_q$.

1: $\mathbf{k}, \widehat{\mathbf{k}} \leftarrow \{0,1\}^m$

2: $\mathsf{ask} \coloneqq \mathbf{k}, \mathsf{apk} \coloneqq (\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{k}), \mathsf{dk} = (\mathbf{A}, \widehat{\mathbf{b}} = \mathbf{A}\widehat{\mathbf{k}})$

3: $(\mathbf{msg}_0, \mathbf{msg}_1, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk}, \mathsf{dk})$

4: Parse $\mathbf{msg}_0 = (\mathsf{msg}_{0,1}, \mathsf{msg}_{0,2})$

5: Parse $\mathbf{msg}_1 = (\mathsf{msg}_{1,1}, \mathsf{msg}_{1,2})$

6: Parse $\mathsf{dk} = (\mathsf{dk}_1, \mathsf{dk}_2)$

7: $\mathbf{s}_1 \leftarrow \mathbb{Z}_q^n, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi^m, \widehat{y}, y_1, y_2 \leftarrow \chi$

8: $\mathsf{dk}_1 = [\mathbf{a}_1, \ldots, \mathbf{a}_m] \in \mathbb{Z}_q^{n \times m}, \mathbf{e}_2^T = [e_{2,1}, \ldots, e_{2,m}] \in \chi^{1 \times m}$

9: **for** $i \in \{1, \ldots, n-1\}$, $x_i \leftarrow \mathbb{Z}_q$

10: **end for**

11: $x_n = (\sum_{i=1}^{n-1} x_i a_{1,i} + \mathbf{s}_1^T \cdot \mathsf{dk}_2 + \widehat{y} + \mathsf{amsg}_1 \cdot \lfloor \frac{q}{2} \rfloor - e_{2,1}) \cdot a_{1,n}^{-1} \mod q$

12: $\mathbf{x}^T = [x_1, \ldots, x_n]$

13: $\mathbf{s}_2 \coloneqq \mathbf{x}$

14: $\mathsf{act}_1^* \coloneqq \mathsf{DReg.Enc}(\mathsf{apk}, \mathsf{msg}_{1,1}; \mathbf{s}_1, \mathbf{e}_1^T, y_1) = (\mathbf{r}_1^T, c_1)$

// $\mathbf{r}_1^T \coloneqq \mathbf{s}_1^T \cdot \mathsf{dk}_1 + \mathbf{e}_1^T$, $c_1 \coloneqq \mathbf{s}_1^T \cdot \mathbf{b} + y_1 + \mathsf{msg}_{1,1} \cdot \lfloor \frac{q}{2} \rfloor \mod q$

15: $\mathsf{act}_2^* \coloneqq \mathsf{DReg.Enc}(\mathsf{pk}, \mathsf{msg}_{1,2}; \mathbf{s}_2, \mathbf{e}_2^T, y_2) = (\mathbf{r}_2^T, c_2)$

// $\mathbf{r}_2^T \coloneqq \mathbf{s}_2^T \cdot \mathsf{dk}_1 + \mathbf{e}_2^T$, $c_2 \coloneqq \mathbf{s}_2^T \cdot \mathbf{b} + y_2 + \mathsf{msg}_{1,2} \cdot \lfloor \frac{q}{2} \rfloor \mod q$

16: **return** $\beta' \leftarrow \mathcal{A}(\mathbf{act}^* = (\mathsf{act}_1^*, \mathsf{act}_2^*))$

The indistinguishability follows the DLWE assumption as in Game 1.

Finally, we observe that Game 3 is exactly the same as $\mathsf{GIND\text{-}CPA}_{\mathsf{AAE}}^{\mathcal{A}}(\lambda, 1)$. This concludes the fully asymmetric security of $\Pi_{\mathsf{aDReg}}$.

**sIND-CPA security.** We prove the strong IND-CPA security by showing that an adversary's view in the sIND-CPA game is computationally indistinguishable from uniformly random. For the sake of contradiction, suppose there exists a PPT adversary $\mathcal{A}$ that breaks down sIND-CPA security of $\Pi_{\mathsf{aDReg}}$ with a non-negligible probability $\sigma$. Then we design a simulator $\mathcal{B}$ that breaks the DLWE assumption.

First, we observe that by the leftover hash lemma $(\mathbf{A}, \mathbf{A} \cdot \widehat{\mathbf{k}} = \widehat{\mathbf{b}})$ is indistinguishable from $(\mathbf{A}, \widehat{\mathbf{b}} \leftarrow \mathbb{Z}_q^n)$ in the $\mathcal{A}$'s view (since $m = \Omega(n \log q)$ and $\widehat{\mathbf{k}}$ is not known to the adversary). Next, by the DLWE assumption, we have

$$((\mathbf{A}||\widehat{\mathbf{b}}),\ \mathbf{s}_1^T \cdot (\mathbf{A}||\widehat{\mathbf{b}}) + (\mathbf{e}_1^T, \widehat{y})) \approx_c ((\mathbf{A}||\widehat{\mathbf{b}}),\ (\mathbf{s}_1^T\mathbf{A} + \mathbf{e}_1^T, u \leftarrow \mathbb{Z}_q) \mod q$$

We specify the simulation as follows:

Simulator $\mathcal{B}(1^\lambda, \mathbf{A}, \widehat{\mathbf{b}}, \mathbf{s}_1^T\mathbf{A} + \mathbf{e}_1^T, u)$

1: $\mathbf{k} \leftarrow \{0,1\}^m$

2: $\mathsf{ask} \coloneqq \mathbf{k}, \mathsf{apk} \coloneqq (\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{k}), \mathsf{dk} = (\mathbf{A}, \widehat{\mathbf{b}})$

3: $(\mathbf{msg}, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk}, \mathsf{ask}, \mathsf{dk})$

4: $\beta \leftarrow \{0,1\}$

5: Parse $\mathsf{dk} = (\mathsf{dk}_1, \mathsf{dk}_2)$

6: Parse $\mathbf{msg} = (\mathsf{msg}_1, \mathsf{msg}_2)$

7: $\mathbf{s}_1 \leftarrow \mathbb{Z}_q^n, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi^m, y_1', y_2 \leftarrow \chi$

8: $\mathsf{dk}_1 = [\mathbf{a}_1, \ldots, \mathbf{a}_m] \in \mathbb{Z}_q^{n \times m}, \mathbf{e}_2^T = [e_{2,1}, \ldots, e_{2,m}] \in \chi^{1 \times m}$

9: **for** $i \in \{1, \ldots, n-1\}$, $x_i \leftarrow \mathbb{Z}_q$

10: **end for**

11: $x_n = (-\sum_{i=1}^{n-1} x_i a_{1,i} + u + \mathsf{amsg}_\beta \cdot \lfloor \frac{q}{2} \rfloor - e_{2,1}) \cdot a_{1,n}^{-1} \mod q$

12: $\mathbf{x}^T = [x_1, \ldots, x_n]$

13: $\mathbf{s}_2 \coloneqq \mathbf{x}$

14: $\mathsf{act}_1^* \coloneqq \mathsf{DReg.Enc}(\mathsf{apk}, \mathsf{msg}_1; \mathbf{s}_1, \mathbf{e}_1^T, y_1 = \mathbf{e}_1^T \cdot \mathbf{k} + y_1') = (\mathbf{r}_1^T, c_1)$

// $\mathbf{r}_1^T \coloneqq \mathbf{s}_1^T \cdot \mathbf{A} + \mathbf{e}_1^T$, $c_1 \coloneqq (\mathbf{s}_1^T\mathbf{A} + \mathbf{e}_1^T)\mathbf{k} + y_1' + \mathsf{msg}_1 \cdot \lfloor \frac{q}{2} \rfloor \mod q$

15: $\mathsf{act}_2^* \coloneqq \mathsf{DReg.Enc}(\mathsf{pk}, \mathsf{msg}_2; \mathbf{s}_2, \mathbf{e}_2^T, y_2) = (\mathbf{r}_2^T, c_2)$

// $\mathbf{r}_2^T \coloneqq \mathbf{s}_2^T \cdot \mathbf{A} + \mathbf{e}_2^T$, $c_2 \coloneqq \mathbf{s}_2^T \cdot \mathbf{b} + y_2 + \mathsf{msg}_2 \cdot \lfloor \frac{q}{2} \rfloor \mod q$

16: $\beta' \leftarrow \mathcal{A}(\mathbf{act}^* = (\mathsf{act}_1^*, \mathsf{act}_2^*))$

17: **return** $\beta == \beta'$

When $u = \mathbf{s}_1^T \cdot \widehat{\mathbf{b}} + \widehat{y}$, then the distribution of $(\mathsf{ask}, \mathsf{apk}, \mathsf{dk})$ and the challenge ciphertext $\mathbf{act}^*$ corresponds to $\mathcal{A}$'s view are the same as in the experiment $\mathsf{GsIND\text{-}CPA}_{\mathsf{AAE}}^{\mathcal{A}}(\lambda, \beta)$. This follows from the anamorphic key-pair being perfectly valid, which makes $\mathbf{act}^*$ a valid encryption of $\mathsf{amsg}_\beta$ under $\mathsf{dk} = (\mathbf{A}, \widehat{\mathbf{b}})$ with $y_1 = \mathbf{e}_1^T \cdot \mathbf{k} + y_1'$ where $y_1'$ is randomly sampled from the LWE error distribution. On the other hand, if $u$ is uniformly random in $\mathbb{Z}_q^n$, then (in line 11) $u + \mathsf{amsg}_\beta \cdot \lfloor \frac{q}{2} \rfloor$ is statistically close to uniform. Therefore, the challenge ciphertext $\mathbf{act}^*$ hide $\mathsf{amsg}_\beta$. Thus, $\mathcal{A}$ has no information on the encrypted anamorphic message. Therefore, if $\mathcal{A}$ breaks the sIND-CPA security of $\Pi_{\mathsf{aDReg}}$ with a non-negligible probability then $\mathcal{B}$ can also break the DLWE assumption with a non-negligible probability. □

### 4.3 Extension of FO-Based PKE

The well-known Fujusaki-Okamoto (FO) transformation [FO13] allows to turn a weakly secure PKE and SKE into a PKE that is IND-CCA secure in the Random Oracle Model. Let $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and $\Pi_{\mathsf{SKE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be the underlying encryption schemes. Then the FO transformation converts them into a scheme $\Pi_{\mathsf{FO}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as follows: $\mathsf{FO.Gen}(1^\lambda)$ simply executes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$. $\mathsf{FO.Enc}(\mathsf{pk}, \mathsf{msg})$ returns $c := (c_1, c_2) := (\mathsf{PKE.Enc}(\mathsf{pk}, r; H(r, \mathsf{msg})), \mathsf{SKE.Enc}(G(r), \mathsf{msg}))$, where $r \leftarrow \mathcal{M}_{\mathsf{PKE}}$ is randomly chosen from the message space of PKE, and $H$ and $G$ are hash functions (modeled as random oracles) mapping to the space of random coins $\mathcal{R}_{\mathsf{PKE}}$ of PKE and the key space $\mathcal{K}_{\mathsf{SKE}}$ of SKE, respectively. To decrypt, $\mathsf{FO.Dec}(\mathsf{sk}, c)$ computes $r := \mathsf{PKE.Dec}(\mathsf{sk}, c_1)$ and returns $\mathsf{SKE.Dec}(G(r), c_2)$ if $c_1 = \mathsf{PKE.Enc}(\mathsf{pk}, r; H(r, \mathsf{msg}))$.

**sIND-RCCA Secure AAE.** The idea to turn $\Pi_{\mathsf{FO}}$ into an AAE scheme is simple: The FO transformation provides us with the recoverable randomness $r \leftarrow \mathcal{M}_{\mathsf{PKE}}$ which we can use to embed the components of a $\Pi_{\mathsf{FO}}$ ciphertext for $\mathsf{amsg}$. For this purpose, the ciphertext components need to be pseudorandom. More precisely, we require IND$-CPA security for $\Pi_{\mathsf{PKE}}$ and wIND$-CPA security for $\Pi_{\mathsf{SKE}}$, both saying that its hard for an adversary to distinguish the encryption of a chosen message from a random ciphertext. In the wIND$-CPA game, no encryption oracle is given. For both definitions, please refer to Section 3. Besides ciphertexts being pseudorandom, they also need to be compatible with $\mathcal{M}_{\mathsf{PKE}}$. To ensure the latter, we require the existence of a pair of PPT algorithms $(\mathsf{Encode}, \mathsf{Decode})$ that allows us to encode ciphertexts from $\mathcal{C}_{\mathsf{FO}} = \mathcal{C}_{\mathsf{PKE}} \times \mathcal{C}_{\mathsf{SKE}}$ to plaintext vectors from $\mathcal{M}_{\mathsf{PKE}}^{\mu}$ for some small $\mu \in \mathbb{N}$ (cf. Definition 23). The resulting $\mu$-message anamorphic extension $\Pi_{\mathsf{FOAE}}$ is presented in Figure 3.

**Definition 23 (Pseudorandom Encodings)** *Let* $\mathsf{Setup}(1^\lambda)$ *be a PPT algorithm that outputs (polynomial-size descriptions of) two sets* $\mathbb{X}$ *and* $\mathbb{Y}$ *which both allow for efficient uniform sampling. Let* $(\mathsf{Encode}, \mathsf{Decode})$ *be a pair of PPT algorithms such that* $\mathsf{Encode}$ *takes an element* $x \in \mathbb{X}$ *as input and outputs an element* $y \in \mathbb{Y}$ *or* $\perp$ *and* $\mathsf{Decode}$ *takes an element* $y \in \mathbb{Y}$ *as input and outputs an element* $x \in \mathbb{X}$ *or* $\perp$*. We call this pair a pseudrandom* $(\mathbb{X}, \mathbb{Y})$*-encoding associated with* $\mathsf{Setup}$*, if it satisfies the following properties:*

1. *Correctness: For any* $(\mathbb{X}, \mathbb{Y}) \leftarrow \mathsf{Setup}(1^\lambda)$*, it holds that*
$$\Pr\left[x \neq x' : \; x \leftarrow \mathbb{X}, \quad y \leftarrow \mathsf{Encode}(x), \quad x' \leftarrow \mathsf{Decode}(y)\right] \leq \mathsf{negl}(\lambda)$$
2. *Pseudorandomness: For any* $(\mathbb{X}, \mathbb{Y}) \leftarrow \mathsf{Setup}(1^\lambda)$*, and any PPT algorithm* $\mathcal{A}$ *holds that*
$$\Pr\left[b = b' : \begin{array}{c} x \leftarrow \mathbb{X}, \; y_0 \leftarrow \mathsf{Encode}(x), \; y_1 \leftarrow \mathbb{Y}, \\ b \leftarrow \{0,1\}, \; b' \leftarrow \mathcal{A}(\mathbb{X}, \mathbb{Y}, y_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

$\underline{\mathsf{aGen}(1^\lambda):}$

1. $(\mathsf{ask}, \mathsf{apk}) \leftarrow \mathsf{FO.Gen}(1^\lambda)$
2. $(\mathsf{tk}', \mathsf{dk}) \leftarrow \mathsf{FO.Gen}(1^\lambda)$
3. $\mathsf{tk} := (\mathsf{ask}, \mathsf{tk}')$
4. **return** $(\mathsf{ask}, \mathsf{apk}, \mathsf{dk}, \mathsf{tk})$

$\underline{\mathsf{aDec}(\mathsf{tk}, \mathbf{act}):}$

1. **for** $j \in \{1, \ldots, \mu\}$:
   Parse $\mathsf{act}_j = (c_1, c_2)$
   $r_j := \mathsf{PKE.Dec}(\mathsf{ask}, c_1)$
   **if** $r_j = \perp$ **return** $\perp$
2. $c := \mathsf{Decode}(r_1, \ldots, r_\mu)$
3. **if** $c = \perp$ **return** $\perp$
4. **return** $\mathsf{FO.Dec}(\mathsf{tk}', c)$

$\underline{\mathsf{aEnc}(\mathsf{apk}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg}):}$

1. Parse $\mathbf{msg} = (\mathsf{msg}_1, \ldots, \mathsf{msg}_\mu)$
2. $c \leftarrow \mathsf{FO.Enc}(\mathsf{dk}, \mathsf{amsg})$
3. $(r_1, \ldots, r_\mu) \leftarrow \mathsf{Encode}(c)$
   // In the negligible event that
   // $\mathsf{Encode}(c) = \perp$, we return to 2.
4. **for** $j \in \{1, \ldots, \mu\}$:
   $\mathsf{act}_j := \mathsf{FO.Enc}(\mathsf{apk}, \mathsf{msg}_j; r_j)$
5. **return act** $:= (\mathsf{act}_1, \ldots, \mathsf{act}_\mu)$

Figure 3: sIND-RCCA Secure $\mu$-AAEE $\Pi_{\mathsf{FOAE}}$

We now present the formal security analysis of our FO-based AAEE scheme described in Figure 3.

**Theorem 3** *Let* $\Pi_{\mathsf{FO}}$ *be an IND-CCA secure FO-based PKE where the underlying PKE* $\Pi_{\mathsf{PKE}}$ *is IND$-CPA secure and has an exponentially large message space* $\mathcal{M}_{\mathsf{PKE}}$ *(i.e.,* $|\mathcal{M}_{\mathsf{PKE}}| \geq 2^{\mathsf{poly}(\lambda)}$ *for non-constant* $\mathsf{poly}$*),*

*and the underlying SKE* $\Pi_{\mathsf{SKE}}$ *is wIND\$-CPA secure. Furthermore, let* $(\mathsf{Encode}, \mathsf{Decode})$ *be a pseudo-random* $(\mathcal{C}_{\mathsf{FO}}, \mathcal{M}^{\mu}_{\mathsf{PKE}})$*-encoding, where* $\mathcal{C}_{\mathsf{FO}}$ *denotes the ciphertext space of* $\Pi_{\mathsf{FO}}$*. Then* $\Pi_{\mathsf{FOAE}}$ *as defined in Figure 3 is a* $\mu$*-message anamorphic extension satisfying anamorphic security and sIND-RCCA security in the ROM.*

**Proof.** We prove this theorem in two parts.

**Anamorphic security.** In the following sequence of hybrid games let $\mathcal{D}$ be a PPT dictator and $Q$ its total number of oracle queries.

Game 0**:** This is the real game $\mathsf{GReal}^{\mathcal{D}}_{\mathsf{FO}}(\lambda)$ in the anamorphic security definition, where the challenger generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$ and handles encryption queries $\mathsf{OEnc}(\mathbf{msg}, \mathsf{amsg})$ of the dictator by computing $(\mathsf{FO.Enc}(\mathsf{pk}, \mathsf{msg}_i)$ for $1 \leq i \leq \mu$. The adversary also has access to the random oracles $H$ and $G$.

Game 1**:** In this game the challenger generates the key pair by computing $(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{FOAE.aGen}(1^\lambda)$ and provides $(\mathsf{apk}, \mathsf{ask})$ as input to $\mathcal{D}$.

Game $(2, j)$ **for** $j \in [0, Q]$**:** In this series of games we slightly modify Game 1. We define Game $(2, 0)$ to be identical to Game 1. In Game $(2, j)$, the challenger handles the first $j$ encryption oracle calls as follows: In such a call, the oracle first randomly chooses an elements $c$ from $\mathcal{C}_{\mathsf{FO}}$ and applies $\mathsf{Encode}(c)$ to obtain $(r_1, \ldots, r_\mu)$. Then the latter values are used as encryption randomness: $(\mathsf{FO.Enc}(\mathsf{apk}, \mathsf{msg}_i; r_i)$ for $1 \leq i \leq \mu$. The remaining oracle calls are left untouched in comparison to Game $(2, 0)$. Note that in Game $(2, Q)$ we have modified all encryption computations in this way.

Game $(3, j)$ **for** $j \in [0, Q]$**:** In this series of games, we modify Game $(2, Q)$. We define Game $(3, 0)$ to be identical to Game $(2, Q)$. In Game $(3, j)$, the challenger handles the first $j$ encryption oracle calls as follows: In such a call, instead of choosing $c$ fully at random from $\mathcal{C}_{\mathsf{FO}}$, we replace the components representing the ciphertext of PKE by an actual encryption of a random message $r'$. That means, we compute $c = (c_{\mathsf{PKE}}, c_{\mathsf{SKE}}) \in \mathcal{C}_{\mathsf{FO}} = \mathcal{C}_{\mathsf{PKE}} \times \mathcal{C}_{\mathsf{SKE}}$ as $c = (\mathsf{PKE.Enc}(\mathsf{dk}, r'), c_{\mathsf{SKE}})$, where $r' \leftarrow \mathcal{M}_{\mathsf{PKE}}$ and $c_{\mathsf{SKE}} \leftarrow \mathcal{C}_{\mathsf{SKE}}$.

Game $(4, j)$ **for** $j \in [0, Q]$**:** In this series of games, we modify Game $(3, Q)$. We define Game $(4, 0)$ to be identical to Game $(3, Q)$. In Game $(4, j)$, the challenger handles the first $j$ encryption oracle calls as follows: In such a call, we now also replace the components of $c$ representing the ciphertext part of SKE. That means, we compute $c$ as $c = (\mathsf{PKE.Enc}(\mathsf{dk}, r'), \mathsf{SKE.Enc}(k, \mathsf{amsg}))$, where $r' \leftarrow \mathcal{M}_{\mathsf{PKE}}$ and $k \leftarrow \mathsf{SKE.Gen}(1^\lambda)$.

Game 5**:** In this game, in all oracle calls when computing $c$, the encryption randomness of PKE and the key $k$ of SKE is generated using the random oracles $H$ and $G$ as in the FO construction, i.e., $c = (\mathsf{PKE.Enc}(\mathsf{dk}, r'; H(r', \mathsf{amsg})), \mathsf{SKE.Enc}(G(r'), \mathsf{amsg}))$, $r' \leftarrow \mathcal{M}_{\mathsf{PKE}}$.

Game 6**:** This is the anamorphic game $\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{FOAE}}(\lambda)$, where the challenger generates $(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{FOAE.aGen}(1^\lambda)$ and handles encryption queries $\mathsf{OaEnc}(\mathbf{msg}, \mathsf{amsg})$ of the dictator by computing the encryption algorithm as $\mathsf{FOAE.aEnc}(\mathsf{apk}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg})$.

Game 0 and Game 1 are identical as $(\mathsf{ask}, \mathsf{apk})$ are generated in aGen by FO.Gen. Game $(2, 0)$ is indistinguishable from Game $(2, Q)$, as a distinguisher $\mathcal{D}$ for Game $(2, j)$ and Game $(2, j + 1)$ (for random $j \in \{0, \ldots, Q - 1\}$) can be turned in to an adversary against the pseudo-randomness of the encoding. Similarly, Game $(3, 0)$ is indistinguishable from Game $(3, Q)$, as from a distinguisher $\mathcal{D}$ for Game $(3, j)$ and Game $(3, j + 1)$ (for random $j$) an IND\$-CPA adversary against $\Pi_{\mathsf{PKE}}$ could be constructed. By a similar hybrid argument, Game $(4, 0)$ is indistinguishable from Game $(4, Q)$, as a distinguisher $\mathcal{D}$ for Game $(4, j)$ and Game $(4, j + 1)$ (for random $j$) leads to an wIND\$-CPA adversary against $\Pi_{\mathsf{SKE}}$. Next, Game $(4, Q)$ and Game 5 are computationally distinguishable: The random oracle call $H(r', \mathsf{amsg})$, for *uniformly chosen* $r'$, will not result in a freshly chosen value if $(r', \mathsf{amsg})$ has been input to $H$ before. If there are $Q_H$ queries to $H$ in total, this could happen with probability at most $\frac{Q_H}{|\mathcal{M}_{\mathsf{PKE}}|}$ for each call to OEnc and thus with probability

$\frac{Q \cdot Q_H}{|\mathcal{M}_{\mathsf{PKE}}|}$ for all calls. Note that our assumption is that the message space is of exponential size. A similar argument holds for random oracle calls $G(r')$. Finally, Game 5 and Game 6 are identical.

**sIND-RCCA Security.** This follows immediately from the IND-CCA security of $\Pi_{\mathsf{FO}}$. The reduction algorithm $\mathcal{B}$ is given $\mathsf{dk}$ as input from its IND-CCA challenger and has access to a $\mathsf{Dec}(\mathsf{tk}', \cdot)$ oracle. It can generate $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$ itself, and thus provide $(\mathsf{apk}, \mathsf{dk}, \mathsf{ask})$ as input to the sIND-RCCA adversary $\mathcal{A}$.

Upon receiving $(\mathsf{msg}, \mathsf{amsg}_0, \mathsf{amsg}_1)$ from $\mathcal{A}$, $\mathcal{B}$ provides $(\mathsf{amsg}_0, \mathsf{amsg}_1)$ to its own IND-CCA challenger to obtain $c^*$. The ciphertext $c^*$ is then transformed into a challenge ciphertext vector $\mathsf{act}^*$ as specified in $\mathsf{aEnc}$ in Figure 3.

To simulate the $\mathsf{aDec}(\mathsf{tk}, \cdot)$ oracle, where $\mathsf{tk} = (\mathsf{ask}, \mathsf{tk}')$, $\mathcal{B}$ uses the self-chosen $\mathsf{ask}$ to decrypt the public key components of $\mathsf{act}$ in order to obtain $(r_1, \ldots, r_\mu)$ (cf. $\mathsf{aDec}$ in Figure 3). Then it decodes them to obtain $c$. If $c = c^*$, $\mathcal{B}$ rejects, i.e., it sends $\perp$ as oracle response to $\mathcal{A}$. Otherwise it queries its own $\mathsf{Dec}(\mathsf{tk}', \cdot)$ to decrypt $c$. If this decryption yields $\mathsf{amsg}_1$ or $\mathsf{amsg}_2$, $\mathcal{B}$ sends $\perp$ as oracle response to $\mathcal{A}$. Note that this simulation of an sIND-RCCA $\mathsf{aDec}$ oracle is perfect: The only thing which is done differently by the simulation is that it checks if $c = c^*$ and returns $\perp$ if this equation holds. However, by correctness of $\Pi_{\mathsf{FO}}$ it follows that such a $c$ would have been decrypted to $\mathsf{amsg}_1$ or $\mathsf{amsg}_2$ and thus the corresponding $\mathsf{act}$ would have also been rejected by the real $\mathsf{aDec}$ oracle. Finally $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. □

**An example instantiation:** Let us consider an instantiation of $\Pi_{\mathsf{FO}}$ based on hashed ElGamal as PKE and XOR for the SKE part. Hashed ElGamal satisfies IND$-CPA security under DDH in the ROM and XORing a message with a random key of the same length satisfies wIND$-CPA security. In this case, $\mathsf{FO.Enc}(\mathsf{pk}, \mathsf{msg})$ would produce a ciphertext $c = (g^{H(r,\mathsf{msg})}, H'(\mathsf{pk}^{H(r,\mathsf{msg})}) \oplus r, G(r) \oplus \mathsf{msg})$, where $H : \{0,1\}^* \times \{0,1\}^* \to \mathbb{Z}_{|\mathbb{G}|}$, $H' : \mathbb{G} \to \{0,1\}^{\ell_1}$, $G : \{0,1\}^* \to \{0,1\}^{\ell_2}$, and $\mathbb{G}$ is the group over which ElGamal is defined. To obtain $\Pi_{\mathsf{FOAE}}$, we also need to instantiate an appropriate pseudorandom encoding. For this, certain parameters need to be fixed: Let us assume $\mathbb{G} = \mathbb{E}(\mathbb{F}_p)$ is an elliptic curve group of prime order $q$, where $p$ is an $\lambda$-bit prime. Furthermore, let us set $\ell_1 = \ell_2 = \lambda$ (but also different parameter settingss like $\ell_1 = \frac{\lambda}{c_1}$ and $\frac{\lambda}{c_1 c_2}$ for $c_1, c_2 \in \mathbb{N}$ s.t. $\ell_1, \ell_2 \in \mathbb{N}$ are possible). In this case, we have to encode random elements of $\mathcal{C}_{\mathsf{FO}} = \mathbb{E}(\mathbb{F}_p) \times \{0,1\}^\lambda \times \{0,1\}^\lambda$ as random elements of $\mathcal{M}^\mu_{\mathsf{PKE}} = (\{0,1\}^\lambda)^\mu$ for some $\mu \in \mathbb{N}$. Obviously, the only difficulty here is encoding the first component, i.e., a random point of $\mathbb{E}(\mathbb{F}_p)$ by one or more random $\lambda$-bit strings. For this purpose, we can deploy Elligator Squared [Tib14]. This is a suitable encoding for essentially arbitrary elliptic curve groups $\mathbb{E}(\mathbb{F}_p)$ (also pairing-friendly) to $\mathbb{F}_p^2$. Encodings of random curve points can be efficiently computed and are statistically indistinguishable from uniformly random elements of $\mathbb{F}_p^2$. Moreover, [Tib14] describes how encodings can be turned into random bit strings. For instance, if $p$ is very close to $2^\lambda$ (which is often the case for standardized curves), and we use the basic $\lambda$-bit representation for elements of $\mathbb{F}_p$ as integers from $\{0, \ldots, p-1\}$, then we can simply interpret the output of the encoding as (statistically close to) uniform bitstrings from $\{0,1\}^\lambda \times \{0,1\}^\lambda$. In this way, we can obtain a pseudorandom encoding of $\mathbb{E}(\mathbb{F}_p) \times \{0,1\}^\lambda \times \{0,1\}^\lambda$ elements as $(\{0,1\}^\lambda)^4$ elements.

**sIND-CCA Secure AAE.** Compared to IND-CCA adversaries, sIND-CCA adversaries are additionally given the secret key $\mathsf{ask}$ of the outer-layer encryption scheme. Considering $\Pi_{\mathsf{FOAE}}$ from Figure 3, they can extract the $r_j$ values from the challenge $\mathbf{act}^*$ and re-embed them into $\mathbf{act} \neq \mathbf{act}^*$ by encrypting different $\mathbf{msg} \neq \mathbf{msg}^*$. Querying the sIND-CCA oracle with $\mathbf{act}$ will reveal $\mathsf{amsg}_\beta$. To protect from this, the idea is to protect the integrity of $\mathbf{act}^*$ by computing a MAC or PRF (which we instantiate by a keyed hash function $T(\cdot)$ in the random oracle model) over $\mathbf{act}^*$ with a key $\mathsf{k}^*_{R,S}$ only shared by the sender and receiver. To generate this key a non-interactive key exchange can be used. The public NIKE key generated by the sender (which is used in combination with the receiver's secret NIKE key to generate $\mathsf{k}^*_{R,S} = \mathsf{k}^*_{S,R}$) as well as the MAC value are embedded as $r_j$ values into additional components of $\mathbf{act}^*$, i.e., we need to extend $\mathbf{act}^*$ compared to Figure 3 to carry these values. Of course, we cannot compute the MAC over the $\mathsf{act}_j$ embedding the MAC value. Note that this approach does not quite protect from the sIND-CCA attack described above, as the adversary could change the MAC key to one it knows by generating and embedding its own public

NIKE sender key and computing the corresponding MAC value over the changed $\mathbf{act} \neq \mathbf{act}^*$, which still contains the inner-layer challenge ciphertext $c^*$. Hence, we need to bind the challenge MAC key $\mathsf{k}^*_{R,S}$ to $c^*$, such that for any $\mathbf{act}$ containing this inner challenge ciphertext only the secret $\mathsf{k}^*_{R,S}$ can be used. This can be done by using a tag-based encryption scheme to generate $c^*$, where the challenge tag is the challenge NIKE public key $\mathsf{kpk}^*_S$ of the sender. Now there is only one problem left: The integrity of the $\mathbf{act}^*$ component containing the MAC value, say $\mathsf{act}^*_\mu$, cannot directly be protected by the MAC. Hence, the adversary might just extract the correct MAC value $r^*_\mu$ and re-encrypt it for a different message $\mathsf{msg}_\mu \neq \mathsf{msg}^*_\mu$ leading to a $\mathbf{act} \neq \mathbf{act}^*$ which can be decrypted by the sIND-CCA oracle. We fix this by including $\mathsf{msg}^*_\mu$ into the MAC computation and by checking that the deterministic encryption of $\mathsf{msg}^*_\mu$ using randomness $r^*_\mu$, which resulted in the outer-layer challenge ciphertext component $\mathsf{act}^*_\mu$, indeed equals the given $\mathsf{act}_\mu$.

The details of our sIND-CCA secure construction $\Pi_{\mathsf{FOAE}^+}$ following the approach just described are shown in Figure 4. We make use of a secure NIKE scheme $\Pi_{\mathsf{NIKE}} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{SharedKey})$ which additionally generates pseudorandom public keys (see Section 3.8), a tag-based encryption scheme $\Pi_{\mathsf{TBE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ which is IND\$-CPA secure and IND-CCA secure (see Section 3.6), and a keyed MAC instantiated as hash function $T$ in the random oracle model. We formally prove that our construction satisfies anamorphic security and sIND-CCA security in Theorem 4.

**Theorem 4** *Let $\Pi_{\mathsf{FO}}$ be FO-based PKE scheme where the underlying PKE $\Pi_{\mathsf{PKE}}$ has an exponentially large message space $\mathcal{M}_{\mathsf{PKE}}$ (i.e., $|\mathcal{M}_{\mathsf{PKE}}| \geq 2^{\mathsf{poly}(\lambda)}$ for non-constant $\mathsf{poly}$) and the underlying SKE scheme is deterministic. Furthermore, let $\Pi_{\mathsf{TBE}}$ be an IND-CCA and IND\$-CPA secure TBE scheme with tag space $\mathcal{T}$, $\Pi_{\mathsf{NIKE}}$ be a secure NIKE scheme with pseudorandom public keys from public key space $\mathcal{PK}_{\mathsf{NIKE}} \subseteq \mathcal{T}$, and $(\mathsf{Encode}, \mathsf{Decode})$ be a pseudorandom $(\mathcal{C}_{\mathsf{TBE}} \times \mathcal{PK}_{\mathsf{NIKE}}, \mathcal{M}^{\mu-1}_{\mathsf{PKE}})$-encoding, where $\mathcal{C}_{\mathsf{TBE}}$ denotes the ciphertext space of $\Pi_{\mathsf{TBE}}$. Lastly, let $T : \{0,1\}^* \to \mathcal{M}_{\mathsf{PKE}}$ be a hash function modeled as random oracle. Then $\Pi_{\mathsf{FOAE}^+}$ as defined in Figure 4 is a $\mu$-message anamorphic extension satisfying anamorphic security and sIND-CCA security in the ROM.*

**Proof.** We prove this theorem in two parts.

**Anamorphic security.** This proof is very similar to the one for Theorem 3. In comparison to the simpler sIND-RCCA secure construction, instead of using $\mathsf{FO.Enc}$ to generate the inner ciphertexts (to be encoded and embedded as FO randomness $r_i$), we now use a TBE, which however, also outputs pseudorandom ciphertexts. Moreover, we have additional components $\mathsf{kpk}_S$ and $T(\cdot)$ to be embedded as $r_i$, which are also pseudorandom.

For completeness, we present the proof, which however, resembles our master theorem in great part. In the following sequence of hybrid games, let $\mathcal{D}$ be a PPT dictator and $Q$ its total number of oracle queries. We use $Q_X$ to denote the number of queries made to a specific oracle $X(\cdot)$.

Game 0**:** This is the game $\mathsf{GReal}^{\mathcal{D}}_{\mathsf{FO}}(\lambda)$ in the anamorphic security definition, where the challenger generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FO.Gen}(1^\lambda)$ and handles encryption queries $\mathsf{OEnc}(\mathbf{msg}, \mathsf{amsg})$ of the dictator by computing $(\mathsf{FO.Enc}(\mathsf{pk}, \mathsf{msg}_i)$ for $1 \leq i \leq \mu$. $\mathcal{D}$ has access to the random oracles $H$, $G$, and $T$.

Game 1**:** In this game the challenger generates the key pair by computing $(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{FOAE}^+.\mathsf{aGen}(1^\lambda)$ and provides $(\mathsf{apk}, \mathsf{ask})$ as input to $\mathcal{D}$.

Game $(2, j)$ **for** $j \in [0, Q]$**:** In this series of games we slightly modify Game 1. We define Game $(2, 0)$ to be identical to Game 1. In Game $(2, j)$, the challenger handles the first $j$ encryption oracle calls as follows: In such a call, the oracle first randomly chooses an element $(c, \mathsf{kpk}_S)$ from $\mathcal{C}_{\mathsf{TBE}} \times \mathcal{PK}_{\mathsf{NIKE}}$ and applies $\mathsf{Encode}(c, \mathsf{kpk}_S)$ to obtain $(r_1, \ldots, r_{\mu-1})$, where $r_\mu$ is still chosen uniformly at random. Then these values are used as encryption randomness: $\mathsf{act}_i = \mathsf{FO.Enc}(\mathsf{apk}, \mathsf{msg}_i; r_i)$ for $1 \leq i \leq \mu$. The remaining oracle calls are left untouched in comparison to Game $(2, 0)$. Note that in Game $(2, Q)$ we have modified all encryption computations in this way.

Game 3**:** We define Game 3 to be identical to Game $(2, Q)$, except that we change the way $r_\mu$ is computed in each oracle call. Instead of choosing $r_\mu$ uniformly at random, we choose $\mathsf{k}_{R,S}$ from the key space of the NIKE scheme and set $r_\mu \coloneqq T(\mathsf{k}_{R,S}, \mathsf{act}_1, \ldots, \mathsf{act}_{\mu-1}, \mathsf{msg}_\mu)$.

$\underline{\mathsf{aGen}(1^\lambda):}$

1. $\mathsf{nikepar} \leftarrow \mathsf{NIKE.Setup}(1^\lambda)$
2. $(\mathsf{kpk}_R, \mathsf{ksk}_R) \leftarrow \mathsf{NIKE.Gen}(\mathsf{nikepar})$
3. $(\mathsf{ask}, \mathsf{apk}) \leftarrow \mathsf{FO.Gen}(1^\lambda)$
4. $(\mathsf{tk}', \mathsf{dk}') \leftarrow \mathsf{TBE.Gen}(1^\lambda)$
5. $\mathsf{dk} \coloneqq (\mathsf{dk}', \mathsf{nikepar}, \mathsf{kpk}_R)$
6. $\mathsf{tk} \coloneqq (\mathsf{ask}, \mathsf{tk}', \mathsf{ksk}_R)$
7. **return** $(\mathsf{ask}, \mathsf{apk}, \mathsf{dk}, \mathsf{tk})$

$\underline{\mathsf{aDec}(\mathsf{tk}, \mathbf{act}):}$

1. **for** $j \in \{1, \ldots, \mu\}$:
   Parse $\mathsf{act}_j = (c_1, c_2)$
   $r_j \coloneqq \mathsf{PKE.Dec}(\mathsf{ask}, c_1)$
   **if** $r_j = \perp$ **return** $\perp$
2. $\mathsf{out} \coloneqq \mathsf{Decode}(r_1, \ldots, r_{\mu-1})$
3. **if** $\mathsf{out} = \perp$ **return** $\perp$
4. parse $(c, \mathsf{kpk}_S) = \mathsf{out}$
5. $\mathsf{k}_{S,R} \leftarrow \mathsf{NIKE.SharedKey}(\mathsf{kpk}_S, \mathsf{ksk}_R)$
6. $\mathsf{msg}_\mu \leftarrow \mathsf{FO.Dec}(\mathsf{ask}, \mathsf{act}_\mu)$
7. **if** $\mathsf{msg}_\mu = \perp$ or $\mathsf{FO.Enc}(\mathsf{apk}, \mathsf{msg}_\mu; r_\mu) \neq \mathsf{act}_\mu$ or $T(\mathsf{k}_{S,R}, \mathsf{act}_1, \ldots, \mathsf{act}_{\mu-1}, \mathsf{msg}_\mu) \neq r_\mu$ **return** $\perp$
8. **return** $\mathsf{TBE.Dec}(\mathsf{tk}', \mathsf{kpk}_S, c)$

$\underline{\mathsf{aEnc}(\mathsf{apk}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg}):}$

1. Parse $\mathbf{msg} = (\mathsf{msg}_1, \ldots, \mathsf{msg}_\mu)$
2. $(\mathsf{kpk}_S, \mathsf{ksk}_S) \leftarrow \mathsf{NIKE.Gen}(\mathsf{nikepar})$
3. $c \leftarrow \mathsf{TBE.Enc}(\mathsf{dk}', \mathsf{kpk}_S, \mathsf{amsg})$
4. $(r_1, \ldots, r_{\mu-1}) \leftarrow \mathsf{Encode}(c, \mathsf{kpk}_S)$
   // In the negligible event that
   // $\mathsf{Encode}(c) = \perp$, we return to 2.
5. **for** $j \in \{1, \ldots, \mu - 1\}$:
   $\mathsf{act}_j \coloneqq \mathsf{FO.Enc}(\mathsf{apk}, \mathsf{msg}_j; r_j)$
6. $\mathsf{k}_{R,S} \leftarrow \mathsf{SharedKey}(\mathsf{kpk}_R, \mathsf{ksk}_S)$
7. $r_\mu \coloneqq T(\mathsf{k}_{R,S}, \mathsf{act}_1, \ldots, \mathsf{act}_{\mu-1}, \mathsf{msg}_\mu)$
8. $\mathsf{act}_\mu \coloneqq \mathsf{FO.Enc}(\mathsf{apk}, \mathsf{msg}_\mu; r_\mu)$
9. **return act** $\coloneqq (\mathsf{act}_1, \ldots, \mathsf{act}_\mu)$

Figure 4: sIND-CCA Secure $\mu$-AAEE $\Pi_{\mathsf{FOAE}^+}$

$\underline{\text{Game 4}}$**:** We define Game 4 to be identical to Game 3, except that we change the way $\mathsf{kpk}_S$ is chosen in each oracle call. Instead of choosing $\mathsf{kpk}_S$ uniformly at random, we generate it using $\mathsf{NIKE.Gen}(\mathsf{nikepar})$.

$\underline{\text{Game 5}}$**:** We define Game 5 to be identical to Game 4, except that we change the way $\mathsf{k}_{R,S}$ is chosen in each oracle call. Instead of choosing it uniformly at random, we generate it using $\mathsf{NIKE.SharedKey}(\mathsf{kpk}_R, \mathsf{ksk}_S)$, where $\mathsf{ksk}_S$ was already generated but not used in Game 4.

$\underline{\text{Game } (6, j)}$ **for** $j \in [0, Q]$**:** In this series of games, we modify Game 5. We define Game $(6, 0)$ to be identical to Game 5. In Game $(6, j)$, the challenger handles the first $j$ encryption oracle calls as follows: In such a call, instead of choosing $c$ at random from $\mathcal{C}_{\mathsf{TBE}}$, we replace it by an actual encryption of $\mathsf{amsg}$: $c \leftarrow \mathsf{TBE.Enc}(\mathsf{dk}', \mathsf{nikepar}_S, \mathsf{amsg})$.

$\underline{\text{Game 7}}$**:** This is the game $\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{FOAE}^+}(\lambda)$, where the challenger generates $(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{FOAE}^+.\mathsf{aGen}(1^\lambda)$ and handles encryption queries $\mathsf{OaEnc}(\mathbf{msg}, \mathsf{amsg})$ of the dictator by computing $\mathsf{FOAE}^+.\mathsf{aEnc}(\mathsf{apk}, \mathsf{dk},$

$\mathbf{msg}, \mathsf{amsg}$).

Game 0 and Game 1 are identical as $(\mathsf{ask}, \mathsf{apk})$ are generated in aGen by FO.Gen. Game $(2,0)$ is indistinguishable from Game $(2,Q)$, as a distinguisher $\mathcal{D}$ for Game $(2,j)$ and Game $(2,j+1)$ (for random $j \in \{0,\ldots,Q-1\}$) can be turned into an adversary against the pseudorandomness of the encoding. Game $(2,Q)$ and Game 3 are statistically indistinguishable with distinguishing advantage at most $Q \cdot Q_T/|\mathsf{k}_{\mathsf{NIKE}}|$ where $|\mathsf{k}_{\mathsf{NIKE}}|$ is the size of the NIKE shared-key space. Since $T$ is a random oracle, $\mathsf{k}_{R,S}$ is information-theoretically hidden from $\mathcal{D}$, the output $T(\mathsf{k}_{R,S},\cdot)$ is uniformly distributed in $\mathcal{D}$'s view, unless $\mathcal{D}$ has previously queried $T$ on some input whose first component equals $\mathsf{k}_{R,S}$. For a fixed oracle call, this collision event happens with probability at most $Q_T/|\mathsf{k}_{\mathsf{NIKE}}|$; over all $Q$ oracle calls, the union bound yields $Q \cdot Q_T/|\mathsf{k}_{\mathsf{NIKE}}|$, which is negligible whenever $|\mathsf{k}_{\mathsf{NIKE}}|$ is super-polynomial. Game 3 and Game 4 are computationally indistinguishable, as a distinguisher could be turned into an adversary against the public-key pseudorandomness property of the NIKE scheme. Game 4 and Game 5 are computationally indistinguishable, as a distinguisher could be turned into an adversary against the security of the NIKE scheme. The indistinguishability of each of these hybrid transitions follows a standard hybrid argument over the $Q$ oracle calls, where each step replaces one uniformly sampled $\mathsf{kpk}_S$ (resp. $\mathsf{k}_{R,S}$) by a NIKE-generated public key (resp. a NIKE shared key) and is indistinguishable by the single-challenge NIKE public-key pseudorandomness (resp. CKS-light security). Game $(6,0)$ is indistinguishable from Game $(6,Q)$, as from a distinguisher $\mathcal{D}$ for Game $(6,j)$ and Game $(6,j+1)$ (for random $j$) an IND\$-CPA adversary against the TBE could be constructed. Finally, Game $(6,Q)$ and Game 7 are identical.

**sIND-CCA Security.** Our strategy is to use the IND-CCA security of $\Pi_{\mathsf{TBE}}$. Consider a reduction algorithm $\mathcal{B}$ who is given $\mathsf{dk}'$ as input from its IND-CCA challenger and has access to a $\mathsf{Dec}(\mathsf{tk}',\cdot,\cdot)$ oracle. It can generate $(\mathsf{apk},\mathsf{ask}) \leftarrow \mathsf{FO.Gen}(1^\lambda)$, as well as $(\mathsf{kpk}_R,\mathsf{ksk}_R) \leftarrow \mathsf{NIKE.Gen}(\mathsf{nikepar})$ itself, and thus provide $(\mathsf{apk},\mathsf{dk},\mathsf{ask})$ as input to the sIND-CCA adversary $\mathcal{A}$ against $\Pi_{\mathsf{FOAE}^+}$.

Upon receiving $(\mathsf{msg}^*,\mathsf{amsg}_0^*,\mathsf{amsg}_1^*)$ from $\mathcal{A}$, the reduction $\mathcal{B}$ samples $(\mathsf{kpk}_S^*,\mathsf{ksk}_S^*) \leftarrow \mathsf{NIKE.Gen}(\mathsf{nikepar})$ itself, and provides $(\mathsf{amsg}_0^*,\mathsf{amsg}_1^*,\mathsf{kpk}_S^*)$ to its own sIND-CCA challenger to obtain the inner-layer challenge ciphertext $c^*$. $(\mathsf{kpk}_S^*,c^*)$ is then transformed into the outer-layer challenge ciphertext vector $\mathbf{act}^*$ as specified in aEnc in Figure 4.

Now the crucial question is how to simulate $\mathcal{A}$'s $\mathsf{aDec}(\mathsf{tk},\cdot)$ oracle, where $\mathsf{tk} = (\mathsf{ask},\mathsf{tk}',\mathsf{ksk}_R)$, using $\mathcal{B}$'s $\mathsf{Dec}(\mathsf{tk}',\cdot,\cdot)$ oracle. Note that $\mathcal{B}$ can use the self-chosen ask to decrypt the public key components of act in order to obtain $(r_1,\ldots,r_\mu)$ (cf. aDec in Figure 4). Then it decodes $(r_1,\ldots,r_{\mu-1})$ to obtain $(c,\mathsf{kpk}_S)$. It can also check whether $T(\mathsf{k}_{S,R},\mathsf{act}_1,\ldots,\mathsf{act}_{\mu-1},\mathsf{msg}_\mu) = r_\mu$ before calling its $\mathsf{Dec}(\mathsf{tk}',\cdot,\cdot)$ oracle on input $(\mathsf{kpk}_S,c)$.

Obviously, the only thing this simulation does differently compared to a real aDec oracle, is calling the $\mathsf{Dec}(\mathsf{tk}',\cdot,\cdot)$ oracle instead of executing the $\mathsf{Dec}(\mathsf{tk}',\cdot,\cdot)$ algorithm. This can lead to a simulation failure if a call to aDec on input of $\mathbf{act} \neq \mathbf{act}^*$ ($\mathbf{act}^*$ would be rejected) would result in calling the $\mathsf{Dec}(\mathsf{tk}',\cdot,\cdot)$ oracle on input $(\mathsf{kpk}_S^*,c^*)$, where the latter is the challenge tag (chosen by $\mathcal{B}$) as well as the challenge $\Pi_{\mathsf{TBE}}$ ciphertext (generated by $\mathcal{B}$'s IND-CCA challenger). In this case the $\mathsf{Dec}(\mathsf{tk}',\cdot,\cdot)$ oracle would reject while the algorithm would decrypt. In the following we will verify that this case cannot occur.

So let us consider the simulation failure event that aDec's input $\mathbf{act} \neq \mathbf{act}^*$ decrypts (via $\mathsf{PKE.Dec}(\mathsf{ask},\cdot)$) to $(r_1,\ldots,r_\mu)$ which are decoded to $(c,\mathsf{kpk}_S) = (c^*,\mathsf{kpk}_S^*)$ and $\mathsf{FO.Enc}(\mathsf{apk},\mathsf{msg}_\mu;r_\mu) = \mathsf{act}_\mu$ as well as $T(\mathsf{k}_{S,R},\mathsf{act}_1,\ldots,\mathsf{act}_{\mu-1},\mathsf{msg}_\mu) = r_\mu$ hold (in which case the $\mathsf{Dec}(\mathsf{tk}',\cdot,\cdot)$ oracle would be called), where $\mathsf{msg}_\mu \leftarrow \mathsf{FO.Dec}(\mathsf{ask},\mathsf{act}_\mu)$ and $\mathsf{k}_{S,R} \leftarrow \mathsf{NIKE.SharedKey}(\mathsf{kpk}_S^*,\mathsf{ksk}_R)$.

The above event decomposes into two mutually exclusive subevents, for which we show that they cannot occur (with overwhelming probability):

1. $(\mathsf{act}_1,\ldots,\mathsf{act}_{\mu-1},\mathsf{msg}_\mu) \neq (\mathsf{act}_1^*,\ldots,\mathsf{act}_{\mu-1}^*,\mathsf{msg}_\mu^*)$: It is important to note that since $(c,\mathsf{kpk}_S) = (c^*,\mathsf{kpk}_S^*)$, the key $\mathsf{k}_{S,R}$ returned by $\mathsf{SharedKey}(\mathsf{kpk}_S^*,\mathsf{ksk}_R)$ in aDec will be the challenge $\mathsf{k}_{S,R} = \mathsf{k}_{S,R}^*$. By the security of the NIKE scheme, $\mathsf{k}_{S,R}^*$ is computationally indistinguishable from random (and thus unknown to $\mathcal{A}$). Hence, $T(\mathsf{k}_{S,R},\mathsf{act}_1,\ldots,\mathsf{act}_{\mu-1},\mathsf{msg}_\mu)$ in aDec, is computed on input of a random $\mathsf{k}_{S,R}$ and since the remaining input to $T$ differs in at least one component to the challenge input, the

$T$ value, being output of the ROM, would need to be guessed by $\mathcal{A}$ and embedded as $r_\mu$ to pass the check. This happens with negligible probability of at most $Q_T/|\mathsf{k}_{\mathsf{NIKE}}| + Q_{\mathsf{aDec}}/|\mathcal{M}_{\mathsf{PKE}}|$, since $\mathsf{k}_{S,R}$ is uniform and hidden from $\mathcal{A}$, making $T(\mathsf{k}_{S,R}, \cdot)$ uniform in $\mathcal{M}_{\mathsf{PKE}}$ in $\mathcal{A}$'s view.

2. $(\mathsf{act}_1, \ldots, \mathsf{act}_{\mu-1}, \mathsf{msg}_\mu) = (\mathsf{act}_1^*, \ldots, \mathsf{act}_{\mu-1}^*, \mathsf{msg}_\mu^*)$: In this case it would need to hold that $\mathsf{act}_\mu \neq \mathsf{act}_\mu^*$ for a simulation failure to occur. Let us consider two subcases. First assume that $r_\mu$ extracted from $\mathsf{act}_\mu$ does not equal $r_\mu^*$. Then also $T(\mathsf{k}_{S,R}^*, \mathsf{act}_1^*, \ldots, \mathsf{act}_{\mu-1}^*, \mathsf{msg}_\mu^*) \neq r_\mu$ and the simulation failure event would not happen. Now assume $r_\mu = r_\mu^*$. But then it must hold that $\mathsf{act}_\mu = \mathsf{act}_\mu^*$ since the equation $\mathsf{FO.Enc}(\mathsf{apk}, \mathsf{msg}_\mu^*; r_\mu^*) = \mathsf{act}_\mu$ is checked by $\mathsf{aDec}$ and $\mathsf{FO.Enc}(\mathsf{apk}, \mathsf{msg}_\mu^*; r_\mu^*)$ is deterministic (since the PKE randomness and the SKE key are deterministically derived as $H(r_\mu^* \mathsf{msg}_\mu^*)$ and $G(r_\mu^*)$, respectively, and the SKE is deterministic by assumption). Hence, this subevent cannot happen.

To summarize, the IND-CCA adversary $\mathcal{B}$ against $\Pi_{\mathsf{TBE}}$ can simulate the $\mathsf{aDec}$ oracle for $\mathcal{A}$ and thus $\mathcal{A}$'s sIND-CCA challenger for $\Pi_{\mathsf{FOAE}^+}$. Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

To make the above argument more formal, one can define a sequence of games, where the first game is the sIND-CCA game for $\Pi_{\mathsf{FOAE}^+}$ for $\beta = 1$ and the last game is the sIND-CCA game for $\Pi_{\mathsf{FOAE}^+}$ for $\beta = 0$. In the second game, we replace $\mathsf{k}_{S,R}^*$ by a random key (which we internally associate with $\mathsf{kpk}_S^*$) by using the security of the NIKE scheme. The next games will modify the $\mathsf{aDec}$ oracle in order to reject in case of the failure event. In the 3rd game, $\mathsf{aDec}$ would reject in case it receives $\mathbf{act} \neq \mathbf{act}*$ leading to the challenge tag $\mathsf{kpk}_S^*$ and $(\mathsf{act}_1, \ldots, \mathsf{act}_{\mu-1}, \mathsf{msg}_\mu) \neq (\mathsf{act}_1^*, \ldots, \mathsf{act}_{\mu-1}^*, \mathsf{msg}_\mu^*)$. Games 2 and 3 can only be distinguished with negligible probability as argued above. In the 4th game, we let $\mathsf{aDec}$ additionally reject if $(\mathsf{act}_1, \ldots, \mathsf{act}_{\mu-1}, \mathsf{msg}_\mu) = (\mathsf{act}_1^*, \ldots, \mathsf{act}_{\mu-1}^*, \mathsf{msg}_\mu^*)$. As argued above, this cannot happen since in this case $\mathbf{act} = \mathbf{act}^*$, so $\mathbf{act}$ would have been rejected before already. So Games 3 and 4 are perfectly indistinguishable. In Game 5, we let $\mathsf{aDec}$ reject as soon as the input $\mathbf{act} \neq \mathbf{act}^*$ leads to the challenge tag $\mathsf{kpk}_S^*$. Clearly, Games 4 and 5 are perfectly indistinguishable. Hence, in Game 5, the modifications we made to $\mathsf{aDec}$ ensure that $\mathsf{TBE.Dec}$ will only be called with a tag $\mathsf{kpk}_S \neq \mathsf{kpk}_S^*$ which is not equal to the challenge tag. Thus, we can now reduce to the security of the TBE, which we do in Game 6, where we encrypt $\mathsf{amsg}_0$ instead of $\mathsf{amsg}_1$. In the remaining games, we reverse the changes we have done before in Games 5 to 2 until we end up with the sIND-CCA game for $\Pi_{\mathsf{FOAE}^+}$ for $\beta = 0$. □

**An example instantiation:** Similar to $\Pi_{\mathsf{FOAE}}$, we can instantiate $\Pi_{\mathsf{FOAE}^+}$ using a hashed ElGamal based FO scheme $\Pi_{\mathsf{FO}}$. We can turn $\Pi_{\mathsf{FO}}$ into a tag-based encryption scheme $\Pi_{\mathsf{TFO}}$ by putting the tag $\mathsf{tag}$ being input to the encryption and decryption algorithm of a TBE into the hash function $H$ for deriving the encryption randomness, i.e., $H(r, \mathsf{msg}, \mathsf{tag})$. $\Pi_{\mathsf{TFO}}$ is IND\$-CPA and IND-CCA secure. The required NIKE can be instantiated by DHKE over $\mathbb{G}$. The pseudorandom encoding function needs to be extended to additionally encode the DH public key using Elligator Squared. Finally, we can use a hash function $T$ outputting $\lambda$-bit strings. This results in a 7-AAEE.

# 5 Identity-Based Anamorphic Encryption Extension

In this section, we formally regard our new concept of reformulating asymmetric anamorphic encryption through the lens of IBE. We further show how to realize this notion using the Boneh-Franklin IBE in Section 5.1. Section 5.2 shows how standardized IBE schemes based on the FO transformation can be turned into anamorphic schemes.

At a high-level, an IBAE is a quadruplet of PPT algorithms $\Pi_{\mathsf{IBAE}} = (\mathsf{aSetup}, \mathsf{aKeyGen}, \mathsf{aEnc}, \mathsf{aDec})$ associated with a regular IBE $\Pi_{\mathsf{IBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ which could be deployed in either of the two modes : *regular* and *anamorphic*. In the regular mode, a dictator computes the master key-pairs $(\mathsf{mpk}, \mathsf{msk})$ using the $\mathsf{Setup}$ algorithm, and publishes the master public key $\mathsf{mpk}$. It then derives user-specific private keys $\mathsf{sk}_{\mathsf{id}}$ based on the public identity $\mathsf{id}$ of users, using the master secret key $\mathsf{msk}$ on the $\mathsf{KeyGen}$ algorithm. A message $\mathsf{msg}$ can be encrypted with respect to $\mathsf{id}$ of a receiver using $\mathsf{mpk}$ on the $\mathsf{Enc}$ algorithm, which can eventually be decrypted using the respective private key $\mathsf{sk}_{\mathsf{id}}$ on $\mathsf{Dec}$ algorithm. As one can clearly see the dictator along with the receiver can recover $\mathsf{msg}$ since it generates the user-specific private keys.

In the anamorphic mode, while the dictator generates the anamorphic master key-pairs $(\mathsf{ampk}, \mathsf{amsk})$ and derives the anamorphic private keys $\mathsf{ask}_{\mathsf{id}}$, the anamorphic clan lead generates the double master key-pairs $(\mathsf{dmpk}, \mathsf{dmsk})$ and the double private key $\mathsf{dsk}_{\mathsf{id}}$ of a user with identity $\mathsf{id}$ who is also an anamorphic clan member. To encrypt a regular message $\mathsf{msg}$ along with an anamorphic message $\mathsf{amsg}$ with respect to $\mathsf{id}$, sender runs $\mathsf{aEnc}$ such that the output ciphertext decrypts to $\mathsf{amsg}$ using $\mathsf{dsk}_{\mathsf{id}}$ on $\mathsf{aDec}$ and $\mathsf{msg}$ using $\mathsf{ask}_{\mathsf{id}}$ on $\mathsf{Dec}$. *Anamorphic security* essentially requires that the dictator should not be able to tell apart the output distributions of regular and anamorphic ciphertexts, along with the corresponding private key pairs. We now proceed more formally.

**Definition 24 ($\mu$-Message IBAE Extension)** *For $\mu \in \mathbb{N}$, a $\mu$-message anamorphic identity-based encryption extension ($\mu$-IBAEE) with an anamorphic message space $\widehat{\mathcal{M}}$ associated with an identity-based encryption scheme $\Pi_{\mathsf{IBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with a regular message space $\mathcal{M}$ is a quadruplet of PPT algorithms $\Pi_{\mathsf{IBAE}} = (\mathsf{aSetup}, \mathsf{aKeyGen}, \mathsf{aEnc}, \mathsf{aDec})$ defined as follows:*

$(\mathsf{ampk}, \mathsf{amsk}, \mathsf{dmpk}, \mathsf{dmsk}) \leftarrow \mathsf{aSetup}(1^\lambda)$ : *The anamorphic key setup algorithm $\mathsf{aSetup}$ takes as input the security parameter $1^\lambda$, and outputs an anamorphic master public key $\mathsf{ampk}$, an anamorphic master secret key $\mathsf{amsk}$, a double master public key $\mathsf{dmpk}$ and a double master secret key $\mathsf{dmsk}$.*

$\mathsf{dsk}_{\mathsf{id}} \leftarrow \mathsf{aKeyGen}(\mathsf{dmsk}, \mathsf{id})$ : *The anamorphic key generation algorithm $\mathsf{aKeyGen}$ takes as input the double master secret key $\mathsf{dmsk}$ and an arbitrary $\mathsf{id} \in \mathcal{ID}$, and outputs an double private key $\mathsf{dsk}_{\mathsf{id}}$ associated with $\mathsf{id}$.*

$\mathbf{act} \leftarrow \mathsf{aEnc}(\mathsf{ampk}, \mathsf{dmpk}, \mathsf{id}, \mathbf{msg}, \mathsf{amsg})$ : *The anamorphic encryption algorithm $\mathsf{aEnc}$ takes as input the anamorphic master public key $\mathsf{ampk}$, double master public key $\mathsf{dmpk}$, an identity $\mathsf{id}$, a regular message vector $\mathbf{msg} \in \mathcal{M}^\mu$, and an anamorphic message $\mathsf{amsg} \in \widehat{\mathcal{M}}$, and outputs an anamorphic ciphertext vector $\mathbf{act}$.*

$\mathsf{amsg} \leftarrow \mathsf{aDec}(\mathsf{ask}_{\mathsf{id}}, \mathsf{dsk}_{\mathsf{id}}, \mathbf{act})$ : *The anamorphic decryption algorithm $\mathsf{aDec}$ takes as input an anamorphic private key $\mathsf{ask}_{\mathsf{id}}$, a double private key $\mathsf{dsk}_{\mathsf{id}}$ and the anamorphic ciphertext vector $\mathbf{act}$, and outputs an anamorphic message $\mathsf{amsg}$ or the symbol $\perp$.*

**Correctness**: For every pair of messages $(\mathbf{msg}, \mathsf{amsg}) \in \mathcal{M}^\mu \times \widehat{\mathcal{M}}$, and for every identity $\mathsf{id} \in \mathcal{ID}$, the following holds:

$$\Pr\left[\mathsf{amsg} = \mathsf{aDec}(\mathsf{ask}_{\mathsf{id}}, \mathsf{dsk}_{\mathsf{id}}, \mathbf{act}) : \begin{array}{c}(\mathsf{ampk}, \mathsf{amsk}, \mathsf{dmpk}, \mathsf{dmsk}) \leftarrow \mathsf{aSetup}(1^\lambda) \\ \mathsf{ask}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{amsk}, \mathsf{id}) \\ \mathsf{dsk}_{\mathsf{id}} \leftarrow \mathsf{aKeyGen}(\mathsf{dmsk}, \mathsf{id}) \\ \mathbf{act} \leftarrow \mathsf{aEnc}(\mathsf{ampk}, \mathsf{dmpk}, \mathsf{id}, \mathbf{msg}, \mathsf{amsg})\end{array}\right] = 1$$

where probability is taken over the random coin tosses of $\mathsf{aSetup}$ and $\mathsf{aEnc}$.

**Anamorphic security**: For every PPT dictator $\mathcal{D}$ it holds that

$$\left|\Pr[\mathsf{GReal}^{\mathcal{D}}_{\mathsf{IBE}}(\lambda) = 1] - \Pr[\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{IBAE}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

where the security games are defined as follows:

$\underline{\mathsf{GReal}^{\mathcal{D}}_{\mathsf{IBE}}(\lambda):}$

1. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
2. **return** $\mathcal{D}^{\mathsf{OEnc}(\cdot,\cdot,\cdot)}(\mathsf{mpk}, \mathsf{msk})$

   where $\mathsf{OEnc}(\mathsf{id}, \mathbf{msg}, \mathsf{amsg}) = \mathbf{ct}$,

   $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{msg}_i)$,

   $\mathbf{msg} = \{\mathsf{msg}_i\}_{i\in[\mu]}, \mathbf{ct} = \{\mathsf{ct}_i\}_{i\in[\mu]}$

$\underline{\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{IBAE}}(\lambda):}$

1. $(\mathsf{ampk}, \mathsf{amsk}, \mathsf{dmpk}, \mathsf{dmsk}) \leftarrow \mathsf{aSetup}(1^\lambda)$
2. **return** $\mathcal{D}^{\mathsf{OaEnc}(\cdot,\cdot,\cdot)}(\mathsf{ampk}, \mathsf{amsk})$

   where $\mathsf{OaEnc}(\mathsf{id}, \mathbf{msg}, \mathsf{amsg}) =$

   $\mathsf{aEnc}(\mathsf{ampk}, \mathsf{dmpk}, \mathsf{id}, \mathbf{msg}, \mathsf{amsg})$

**Fully asymmetric IND-CPA security**: An IBAE satisfies fully asymmetric IND-CPA security if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\left|\Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBAE}}(\lambda, 0) = 1] - \Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBAE}}(\lambda, 1) = 1]\right| \leq \mathsf{negl}(\lambda)$$

where the security game is defined as follows:

$\underline{\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBAE}}(1^\lambda, \beta):}$

1. $(\mathsf{ampk}, \mathsf{amsk}, \mathsf{dmpk}, \mathsf{dmsk}) \leftarrow \mathsf{aSetup}(1^\lambda)$
2. $(\mathsf{id}^*, \mathbf{msg}_0, \mathbf{msg}_1, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}^{\mathsf{OKG}(\cdot)}(\mathsf{ampk}, \mathsf{dmpk})$
3. $\mathbf{act}_\beta = \mathsf{aEnc}(\mathsf{ampk}, \mathsf{dmpk}, \mathsf{id}^*, \mathbf{msg}_\beta, \mathsf{amsg}_\beta)$
4. **return** $\beta' \leftarrow \mathcal{A}^{\mathsf{OKG}(\cdot)}(\mathbf{act}_\beta)$

   where $\mathsf{OKG}(\mathsf{id}) = (\mathsf{KeyGen}(\mathsf{amsk}, \mathsf{id}), \mathsf{aKeyGen}(\mathsf{dmsk}, \mathsf{id}))$

   and $\mathsf{id}^*$ is not queried to $\mathsf{OKG}(\cdot)$.

We also consider additional security definitions for IBAE.

**Strong IND-CPA security and beyond**: An IBAE satisfies strong IND-CPA security if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\left|\Pr[\mathsf{GsIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBAE}}(\lambda, 0) = 1] - \Pr[\mathsf{GsIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBAE}}(\lambda, 1) = 1]\right| \leq \mathsf{negl}(\lambda)$$

where the security game is defined as follows:

$\underline{\mathsf{GsIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBAE}}(1^\lambda, \beta):}$

1. $(\mathsf{ampk}, \mathsf{amsk}, \mathsf{dmpk}, \mathsf{dmsk}) \leftarrow \mathsf{aSetup}(1^\lambda)$
2. $(\mathsf{id}^*, \mathbf{msg}, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}^{\mathsf{OaKeyGen}(\cdot)}(\mathsf{ampk}, \mathsf{amsk}, \mathsf{dmpk})$
3. $\mathbf{act}_\beta = \mathsf{aEnc}(\mathsf{ampk}, \mathsf{dmpk}, \mathsf{id}^*, \mathbf{msg}, \mathsf{amsg}_\beta)$
4. **return** $\beta' \leftarrow \mathcal{A}^{\mathsf{OaKeyGen}(\cdot)}(\mathbf{act}_\beta)$

   where $\mathsf{OaKeyGen}(\mathsf{id}) = \mathsf{aKeyGen}(\mathsf{dmsk}, \mathsf{id})$

   and $\mathsf{id}^*$ is not queried to $\mathsf{OaKeyGen}(\cdot)$.

We call $\Pi_{\mathsf{IBAE}}$ fully asymmetric IND-CCA secure if the advantage of $\mathcal{A}$ in $\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBAE}}$ is negligible when it is also given access to an oracle $\mathsf{ODec}(\cdot,\cdot)$ that on input $(\mathsf{id}, \mathsf{act})$ returns $(\mathsf{Dec}(\mathsf{amsk}, \mathsf{id}, \mathbf{act}), \mathsf{aDec}(\mathsf{dmsk}, \mathsf{id}, \mathbf{act}))$, which decrypts every given $\mathsf{act}$ except for the challenge ciphertext $\mathbf{act}_\beta$ with the challenge identity $\mathsf{id}^*$. To also guarantee IND-CCA security with respect to the dictator who knows $\mathsf{ask}$, we additionally provide $\mathsf{ask}$ to $\mathcal{A}$ as input and call the corresponding security notion strong IND-CCA secure (sIND-CCA) where $\mathsf{ODec}(\cdot,\cdot)$ is replaced with the oracle $\mathsf{aDec}(\mathsf{dmsk}, \cdot, \cdot)$. Since, unfortunately, our constructions do not achieve sIND-CCA security, we define a relaxation in the spirit of Replayable CCA (RCCA) security for IBE similar to the case of PKE: RCCA captures a security notion which is identical to CCA except for the added ability of the attacker to generate new ciphertexts that decrypt to the same plaintexts as previously seen ciphertexts. More precisely, our sIND-RCCA game coincides with the sIND-CCA game, except that the $\mathsf{aDec}(\mathsf{dmsk}, \cdot, \cdot)$-oracle rejects for a given $(\mathsf{id}, \mathsf{act})$ if $\mathsf{aDec}(\mathsf{dmsk}, \mathsf{id}, \mathbf{act})$ results in $\mathsf{amsg}_0$ or $\mathsf{amsg}_1$.

**sIND-RCCA security**: An IBAE satisfies sIND-RCCA security if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\left|\Pr[\mathsf{GsIND\text{-}RCCA}^{\mathcal{A}}_{\mathsf{IBAE}}(\lambda, 0) = 1] - \Pr[\mathsf{GsIND\text{-}RCCA}^{\mathcal{A}}_{\mathsf{IBAE}}(\lambda, 1) = 1]\right| \leq \mathsf{negl}(\lambda)$$

where the security game is defined as follows:

$\underline{\mathsf{GsIND\text{-}RCCA}^{\mathcal{A}}_{\mathsf{IBAE}}(1^\lambda, \beta):}$

1. $(\mathsf{ampk}, \mathsf{amsk}, \mathsf{dmpk}, \mathsf{dmsk}) \leftarrow \mathsf{aSetup}(1^\lambda)$
2. $(\mathsf{id}^*, \mathbf{msg}, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}^{\mathsf{OaKeyGen}(\cdot), \mathsf{OaDec}(\cdot,\cdot)}(\mathsf{ampk}, \mathsf{amsk}, \mathsf{dmpk})$
3. $\mathbf{act}_\beta = \mathsf{aEnc}(\mathsf{ampk}, \mathsf{dmpk}, \mathsf{id}^*, \mathbf{msg}, \mathsf{amsg}_\beta)$
4. **return** $\beta' \leftarrow \mathcal{A}^{\mathsf{OaKeyGen}(\cdot), \mathsf{OaDec}(\cdot,\cdot)}(\mathbf{act}_\beta)$

*where* $\mathsf{OaKeyGen}(\mathsf{id}) = \mathsf{aKeyGen}(\mathsf{dmsk}, \mathsf{id})$,

$\mathsf{OaDec}(\mathsf{id}, \mathbf{act}) = \mathsf{aDec}(\mathsf{dmsk}, \mathsf{id}, \mathbf{act})$

*and* $\mathsf{id}^*$ is not queried to $\mathsf{OaKeyGen}(\cdot)$

*and* $(\mathsf{id}^*, \mathbf{act}_\beta)$ is not queried to $\mathsf{OaDec}(\cdot,\cdot)$

*and* $\mathsf{OaDec}(\mathsf{id}, \mathbf{act})$ outputs $\perp$ if $\mathsf{aDec}(\mathsf{dmsk}, \mathsf{id}, \mathbf{act}) = \mathsf{amsg}_0$ or $\mathsf{amsg}_1$.

We now formally regard the IBAE-compatible notion for an IBE. Building upon our previous insight, the notion captures *two* properties: *on-the-fly double key generation* property simply points out that the double master key pairs $(\mathsf{dmpk}, \mathsf{dmsk})$ can simply be derived from the regular IBE master key pairs $(\mathsf{mpk}, \mathsf{msk})$, while the *ciphertext extractibility* property invariably indicates that $\mathsf{IBAE.aDec}$ is essentially the same as $\mathsf{IBE.Dec}$ under the existence of an efficiently computable function $F$. More explicitly, $F(\mathsf{ask}_{\mathsf{id}}, \mathbf{act})$ produces a valid ciphertext that extracts to $\mathsf{amsg}$ using $\mathsf{dsk}_{\mathsf{id}}$ on $\mathsf{IBE.Dec}$.

**Definition 25 (IBAE-Compatible)** *An identity-based encryption scheme* $\Pi_{\mathsf{IBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be IBAE-compatible for the quadruplet* $\Pi_{\mathsf{IBAE}} = (\mathsf{aSetup}, \mathsf{aKeyGen}, \mathsf{aEnc}, \mathsf{aDec})$ *if the following holds:*

- *On-the-fly double key generation: A part of output* $(\mathsf{dmpk}, \mathsf{dmsk})$ *of* $\mathsf{IBAE.aSetup}(1^\lambda)$ *is the output of* $\mathsf{IBE.Setup}(1^\lambda)$. *In particular, if* $\mathsf{IBAE.aSetup}(1^\lambda) = (\mathsf{ampk}, \mathsf{amsk}, \mathsf{dmpk}, \mathsf{dmsk})$ *and* $\mathsf{IBE.Setup}(1^\lambda) = (\mathsf{mpk}, \mathsf{msk})$, $\mathsf{IBE.Setup}(1^\lambda) = (\mathsf{mpk}', \mathsf{msk}')$ *then* $\mathsf{ampk} = \mathsf{mpk}, \mathsf{amsk} = \mathsf{msk}, \mathsf{dmsk} = \mathsf{msk}'$ *and there exists an efficiently computable function* $G$ *such that* $\mathsf{dmpk} = G(\mathsf{mpk}, \mathsf{msk}, \mathsf{mpk}', \mathsf{msk}')$[7].

[7] In $\mathsf{dmpk} = G(\mathsf{mpk}, \mathsf{msk}, \mathsf{mpk}', \mathsf{msk}')$, the $\mathsf{msk}$ could be $\perp$ in certain scenarios (see Remark 2).

- *Ciphertext extractability: There exists an efficiently computable function $F$ such that* $\mathsf{IBE.Dec}(\mathsf{dsk_{id}}, F(\mathsf{ask_{id}}, \mathbf{act})) = \mathsf{amsg}$*, where* $\mathbf{act} = \mathsf{IBAE.aEnc}(\mathsf{ampk}, \mathsf{dmpk}, \mathsf{id}, \mathbf{msg}, \mathsf{amsg})$ *for an* $\mathsf{id} \in \mathcal{ID}$*.*

**Remark 2** The on-the-fly double key generation property guarantees that the double master secret key $\mathsf{dmsk}$ can be independently sampled via the $\mathsf{IBE.Setup}$ algorithm, without requiring knowledge of the original key pair $(\mathsf{ampk}, \mathsf{amsk})$. Moreover, the corresponding double master public key $\mathsf{dmpk}$ can be efficiently derived from two independently generated key pairs: the original $(\mathsf{mpk} = \mathsf{ampk}, \mathsf{msk} = \mathsf{amsk})$ and a freshly sampled $(\mathsf{mpk}', \mathsf{msk}' = \mathsf{dmsk})$. This property supports two operational paradigms:

1. **Authority-before-dictatorship:** Here, the dictator initially obtains the key pair $(\mathsf{mpk}, \mathsf{msk})$ by coercing the key generation authority of an IBE system. Subsequently, the original authority must be able to generate a valid double master public key $\mathsf{dmpk} = G(\mathsf{mpk}, \mathsf{msk}, \mathsf{mpk}', \mathsf{msk}')$ by independently executing $\mathsf{IBE.Setup}$ and becomes an anamorphic clan leader.

2. **Authority-after-dictatorship:** In this setting—preferred and further motivated in Section 2—the dictator independently generates the key pair $(\mathsf{mpk}, \mathsf{msk})$, and any party can later assume the role of the anamorphic clan leader by independently sampling $(\mathsf{msk}', \mathsf{mpk}')$ through $\mathsf{IBE.Setup}$ and computing the joint public key as $\mathsf{dmpk} = G(\mathsf{mpk}, \perp, \mathsf{mpk}', \mathsf{msk}' = \mathsf{dmsk})$.

Our construction of IBAEE, instantiated from the Boneh-Franklin $\mathsf{BasicIdent}$ scheme (Section 5.1), conforms to the second paradigm. In this case, the double master public key $\mathsf{dmpk}$ can be deterministically computed using the original authority's public key $\mathsf{ampk}$ and the independently generated secret key $\mathsf{dmsk}$, aligning with the on-the-fly generation requirement.

## 5.1 Extension of the Boneh-Franklin BasicIdent Scheme

We now present an IBAE construction based on the seminal Boneh-Franklin basic IBE scheme [BF01]. We focus on a simple version that encrypts single bits. The $\mathsf{BasicIdent}$ identity-based encryption scheme [BF01] is a quadruplet of PPT algorithms $\Pi_{\mathsf{BF}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined over a security parameter $\lambda \in \mathbb{N}$ as follows: let prime $q > 2^\lambda$ be a polynomial in $\lambda$. The setup algorithm $\mathsf{Setup}$ on input $1^\lambda$ samples groups $\mathbb{G}$ and $\mathbb{G}_T$ of order $q$ and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, along with an arbitrary group generator $P \in \mathbb{G}$. Define $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}^*$, $\mathsf{H}_2 : \mathbb{G}_T \to \{0,1\}$. Then for a uniformly random $s$ in $\mathbb{Z}_q^*$, compute $P_{\mathsf{pub}} \coloneqq P^s$. The algorithm finally returns a master secret key $\mathsf{msk} \coloneqq s$ and a master public key $\mathsf{mpk} \coloneqq (\mathbb{G}, \mathbb{G}_T, P, e, q, P_{\mathsf{pub}}, \mathsf{H}_1, \mathsf{H}_2)$. The key generation algorithm $\mathsf{KeyGen}$ on input $\mathsf{id} \in \{0,1\}^*$ and $\mathsf{msk}$, computes $Q_{\mathsf{id}} \coloneqq \mathsf{H}_1(\mathsf{id})$, and returns a private key $\mathsf{sk}_{id} \coloneqq Q_{\mathsf{id}}^s$ corresponding to $\mathsf{id}$. To encrypt a message $\mathsf{msg} \in \mathcal{M}$ towards a public identity $\mathsf{id}$, where $\mathcal{M} = \{0,1\}$, the encryption algorithm $\mathsf{Enc}$ samples a uniformly random $\kappa$ in $\mathbb{Z}_q^*$ and returns ciphertext $\mathsf{ct}$ as $(\mathsf{ct}_1, \mathsf{ct}_2) = (P^\kappa, \mathsf{msg} \oplus \mathsf{H}_2(g_{\mathsf{id}}^\kappa))$, where $g_{\mathsf{id}} = e(Q_{\mathsf{id}}, P_{\mathsf{pub}})$. The decryption algorithm $\mathsf{Dec}$ simply recovers $\mathsf{msg}$ by computing $\mathsf{ct}_2 \oplus \mathsf{H}_2(e(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}_1))$. The $\mathsf{BasicIdent}$ scheme is known to be IND-CPA secure under the $\mathsf{DBDH}$ assumption for $\mathbb{G}$.

**Boneh-Franklin 2-Message IBAE.** The Boneh-Franklin 2-IBAE extension $\Pi_{\mathsf{aBF}} = (\mathsf{aSetup}, \mathsf{aKeyGen}, \mathsf{aEnc}, \mathsf{aDec})$ works as follows: the anamorphic key generation algorithm $\mathsf{aBF.aSetup}$ initially sets $(\mathsf{ampk}, \mathsf{amsk}) = (\mathsf{mpk}, \mathsf{msk})$ as in the regular $\mathsf{BF.Setup}$, following which it samples a uniformly random $\alpha \in \mathbb{Z}_q^*$, and sets the double master secret key $\mathsf{dmsk} \coloneqq \alpha$ and the double master public key $\mathsf{dmpk} \coloneqq P^\alpha$. The anamorphic key generation algorithm $\mathsf{aBF.aKeyGen}$ takes as input $\mathsf{dmsk}$ and the user identity $\mathsf{id}$ and generates the double private key $\mathsf{dsk_{id}}$ associated with $\mathsf{id}$. On input a regular message vector $\mathbf{msg} = (\mathsf{msg}_1, \mathsf{msg}_2) \in \mathcal{M}^2$, and an anamorphic message $\mathsf{amsg} \in \widehat{\mathcal{M}}$, where $\mathcal{M} = \widehat{\mathcal{M}} = \{0,1\}$, the anamorphic encryption algorithm $\mathsf{aBF.aEnc}$ computes $\mathsf{act}_1 = (r_1, c_1)$ using a uniformly random $\kappa_1$ on $\mathsf{msg}_1$ as $\mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_1, \mathsf{id}; \kappa_1)$. However, for encoding $\mathsf{amsg}$ in $c_2$ (second component of $\mathsf{act}_2$), the algorithm chooses $\kappa_2$ subject to the constraint $\mathsf{H}_2(g_{\mathsf{id}}^{\kappa_2}) = \mathsf{amsg} \oplus \mathsf{H}_2(e(Q_{\mathsf{id}}, \mathsf{ampk})^{\kappa_1})$ and computes $\mathsf{act}_2 = (r_2, c_2)$ as $\mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_2, \mathsf{id}; \kappa_2)$. Finally, the anamorphic ciphertext vector is $\mathbf{act} = (\mathsf{act}_1, \mathsf{act}_2)$. To recover $\mathsf{amsg}$, the anamorphic decryption algorithm $\mathsf{aBF.aDec}$ first extracts $\mathsf{H}_2(g_{\mathsf{id}}^{\kappa_2})$ by computing $\mathsf{H}_2(e(\mathsf{ask}_{\mathsf{id}}, r_2))$. Note that $\mathsf{H}_2(g_{\mathsf{id}}^{\kappa_2})$ is set to be equal to $\mathsf{amsg} \oplus \mathsf{H}_2(e(Q_{\mathsf{id}}, \mathsf{ampk})^{\kappa_1})$, call it $\widehat{c}$. Then it computes $\mathsf{BF.Dec}$ on $(\widehat{r} = r_1, \widehat{c})$. Recall that the dictator has

no information on the double private key $\mathsf{dsk}_{\mathsf{id}}$ associated with $\mathsf{id}$, and hence is oblivious to the presence of $\mathsf{amsg}$. We present our construction in Figure 5.

$\underline{\mathsf{aSetup}(1^\lambda):}$

1. $(\mathsf{mpk} := (\mathbb{G}, \mathbb{G}_T, P, e, q, P_{\mathsf{pub}}, \mathsf{H}_1, \mathsf{H}_2),\ \mathsf{msk} := s) \leftarrow \mathsf{BF.Setup}(1^\lambda)$
2. $\alpha \leftarrow \mathbb{Z}_q^*$
3. $\mathsf{ampk} := \mathsf{mpk}, \mathsf{amsk} := \mathsf{msk}$
4. $\mathsf{dmpk} := P^\alpha, \mathsf{dmsk} := \alpha$
5. **return** $(\mathsf{ampk}, \mathsf{amsk}, \mathsf{dmpk}, \mathsf{dmsk})$

$\underline{\mathsf{aKeyGen}(\mathsf{dmsk}, \mathsf{id}):}$

1. $Q_{\mathsf{id}} := \mathsf{H}_1(\mathsf{id})$
2. $\mathsf{dsk}_{\mathsf{id}} := Q_{\mathsf{id}}^\alpha$
3. **return** $\mathsf{dsk}_{\mathsf{id}}$

$\underline{\mathsf{aDec}(\mathsf{ask}_{\mathsf{id}}, \mathsf{dsk}_{\mathsf{id}}, \mathbf{act}):}$

1. $\mathsf{amsg} := \mathsf{BF.Dec}(\mathsf{dsk}_{\mathsf{id}}, F(\mathsf{ask}_{\mathsf{id}}, \mathbf{act}))$
2. **return** $\mathsf{amsg}$

$\underline{\mathsf{aEnc}(\mathsf{ampk}, \mathsf{dmpk}, \mathsf{id}, \mathbf{msg}, \mathsf{amsg}):}$

1. Parse $\mathbf{msg} = (\mathsf{msg}_1, \mathsf{msg}_2)$
2. $Q_{\mathsf{id}} := \mathsf{H}_1(\mathsf{id})$
3. $\widehat{g}_{\mathsf{id}} := e(Q_{\mathsf{id}}, \mathsf{dmpk}),\ g_{\mathsf{id}} := e(Q_{\mathsf{id}}, P_{\mathsf{pub}})$
4. $\kappa_1, \kappa_2 \leftarrow \mathbb{Z}_q^*$ s.t. $\mathsf{H}_2(g_{\mathsf{id}}^{\kappa_2}) = \mathsf{amsg} \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}}^{\kappa_1})$
5. $\mathsf{act}_1 := \mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_1, \mathsf{id}; \kappa_1)$

   // $\mathsf{act}_1 = (r_1, c_1)$

   // $r_1 = P^{\kappa_1}, c_1 = \mathsf{msg}_1 \oplus \mathsf{H}_2(g_{\mathsf{id}}^{\kappa_1})$
6. $\mathsf{act}_2 := \mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_2, \mathsf{id}; \kappa_2)$

   // $\mathsf{act}_2 = (r_2, c_2)$

   // $r_2 = P^{\kappa_2}, c_2 = \mathsf{msg}_2 \oplus \mathsf{H}_2(g_{\mathsf{id}}^{\kappa_2})$
7. **return** $\mathbf{act} := (\mathsf{act}_1, \mathsf{act}_2)$

$\underline{F(\mathsf{ask}_{\mathsf{id}}, \mathbf{act}):}$

1. Parse $\mathbf{act} = (\mathsf{act}_1, \mathsf{act}_2)$
2. Parse $\mathsf{act}_1 = (\mathbf{r}_1, c_1)$,

   $\mathsf{act}_2 = (\mathbf{r}_2, c_2)$
3. $\mathsf{H}_2(e(\mathsf{ask}_{\mathsf{id}}, r_2)) = \mathsf{H}_2(g_{\mathsf{id}}^{\kappa_2}) = \mathsf{amsg} \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}}^{\kappa_1})$

   // $\mathsf{ask}_{\mathsf{id}} \leftarrow \mathsf{BF.KeyGen}(\mathsf{amsk}, \mathsf{id})$
4. $\widehat{r} := r_1,\ \widehat{c} := \mathsf{amsg} \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}}^{\kappa_1})$
5. **return** $(\widehat{r}, \widehat{c})$

Figure 5: 2-IBAEE $\Pi_{\mathsf{aBF}}$

**Correctness, Compatibility and Security Analysis.** We provide proofs of correctness, compatibility, anamorphic security and (s)IND-CPA security of $\Pi_{\mathsf{aBF}}$ below.

**Correctness**: We now show the correctness of the $\Pi_{\mathsf{aBF}} = (\mathsf{aSetup}, \mathsf{aKeyGen}, \mathsf{aEnc}, \mathsf{aDec})$ scheme based on the Boneh-Franklin IBE scheme. Let $\mathbf{act} = (\mathsf{act}_1 \mathsf{act}_2)$ be the output of $\mathsf{aEnc}(\mathsf{ampk}, \mathsf{dmpk}, \mathsf{id}, \mathbf{msg}, \mathsf{amsg})$, where $\mathsf{act}_1 = (r_1, c_1)$ and $\mathsf{act}_2 = (r_2, c_2)$. Here, $r_1 = P^{\kappa_1}$ and $r_2 = P^{\kappa_2}$ are generated using a uniformly random $\kappa_1$ in $\mathbb{Z}_q^*$, while $\kappa_2$ is such that $\mathsf{H}_2(g_{\mathsf{id}}^{\kappa_2}) = \mathsf{amsg} \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}}^{\kappa_1})$, where $\widehat{g}_{\mathsf{id}} := e(Q_{\mathsf{id}}, \mathsf{dmpk} = P^\alpha)$ and $g_{\mathsf{id}} := e(Q_{\mathsf{id}}, P_{\mathsf{pub}} = P^s)$. The anamorphic decryption algorithm $\mathsf{aBF.aDec}$ knows both the anamorphic private key $\mathsf{ask}_{\mathsf{id}} = Q_{\mathsf{id}}^s$ and the double private key $\mathsf{dsk}_{\mathsf{id}} = Q_{\mathsf{id}}^\alpha$. It first calls $F(\mathsf{ask}_{\mathsf{id}}, \mathbf{act})$, which extracts $\mathsf{H}_2(g_{\mathsf{id}}^{\kappa_2})$ by computing $\mathsf{H}_2(e(\mathsf{ask}_{\mathsf{id}}, r_2))$. Note that $e(\mathsf{ask}_{\mathsf{id}}, r_2) = e(Q_{\mathsf{id}}^s, P^{\kappa_2}) = e(Q_{\mathsf{id}}, P^s)^{\kappa_2} = (g_{\mathsf{id}}^{\kappa_2})$. Finally,

$\mathsf{amsg}$ is recovered by executing $\mathsf{BF.Dec}$ on $\widehat{r} = r_1$ and $\widehat{c} = \mathsf{H}_2(g_{\mathsf{id}}^{\kappa_2}) = \mathsf{amsg} \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}}^{\kappa_1})$. To see why it works, observe that $\mathsf{BF.Dec}(\mathsf{dsk}_{\mathsf{id}}, F(\mathsf{ask}_{\mathsf{id}}, \mathbf{act}))$ computes $\widehat{c} \oplus \mathsf{H}_2(e(\mathsf{dsk}_{\mathsf{id}}, \widehat{r})) = \mathsf{amsg} \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}}^{\kappa_1}) \oplus \mathsf{H}_2(e(Q_{\mathsf{id}}^{\alpha}, P^{\kappa_1})) = \mathsf{amsg} \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}}^{\kappa_1}) \oplus \mathsf{H}_2(e(Q_{\mathsf{id}}, P^{\alpha})^{\kappa_1}) = \mathsf{amsg}$.

**Lemma 4** *Boneh-Franklin is IBAE-compatible for* $\Pi_{\mathsf{aBF}}$ *described in Figure 5.*

**Proof.** First, observe that $\mathsf{aBF.aSetup}(1^\lambda)$ outputs $\mathsf{amsk} = s$, $\mathsf{ampk} = (\mathbb{G}, \mathbb{G}_T, P, e, q, P_{\mathsf{pub}} = P^s, \mathsf{H}_1, \mathsf{H}_2)$, $\mathsf{dmsk} = \alpha$ and $\mathsf{dmpk} = P^\alpha$, where $s$ and $\alpha$ are uniformly random in $\mathbb{Z}_q^*$. Therefore, the key pair $(\mathsf{ampk}, \mathsf{amsk})$ has identical distributions with $\{(\mathsf{mpk}, \mathsf{msk}) : \mathsf{msk} = \widetilde{s}, \mathsf{mpk} = (\mathbb{G}', \mathbb{G}_T', P', e', q', P_{\mathsf{pub}}' = P^{\widetilde{s}}, \mathsf{H}_1', \mathsf{H}_2')$ such that $\widetilde{s} \leftarrow \mathbb{Z}_q^*$, which is essentially the distribution of master secret and master public keys of a Boneh-Franklin IBE scheme. Additionally, it is easy to see that the double key pair $(\mathsf{dmpk}, \mathsf{dmsk})$ can be efficiently obtained from the knowledge of only $\mathsf{ampk}$. Hence, the IBAE satisfies *on-the-fly double key generation* property. Next, observe that the function $F$ defined in Figure 5 takes as input $\mathbf{act} \leftarrow \mathsf{aBF.aEnc}(\mathsf{ampk}, \mathsf{dmpk}, \mathsf{id}, \mathbf{msg}, \mathsf{amsg})$, and parses it as $\mathsf{act}_1 = (r_1, c_1)$ and $\mathsf{act}_2 = (r_2, c_2)$. It then extracts $r_1$ (call it $\widehat{r}$) and the component $\mathsf{H}_2(g_{\mathsf{id}}^{\kappa_2}) = \mathsf{amsg} \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}}^{\kappa_1})$(call it $\widehat{c}$). Note that $(\widehat{\mathbf{r}}, \widehat{c})$ is a valid Boneh-Franklin ciphertext as $\widehat{r} = r_1$ and $\widehat{c} = \mathsf{amsg} \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}}^{\kappa_1})$. Thus, $\mathsf{amsg}$ can be extracted by executing $\mathsf{BF.Dec}$ using $\mathsf{dsk}_{\mathsf{id}} = Q_{\mathsf{id}}^{\alpha}$, which implies that the IBAE satisfies the *ciphertext extractibility* property. Hence we conclude that the Boneh-Franklin IBE is IBAE-compatible. □

**Theorem 5** *Under the* $\mathsf{DBDH}$ *assumption for* $(\mathbb{G}, \mathbb{G}_T, P, e, q)$*, the Boneh-Franklin* 2*-message identity-based anamorphic encryption extension* $\Pi_{\mathsf{aBF}}$ *described in Figure 5 satisfies anamorphic, fully asymmetric IND-CPA and sIND-CPA security.*

**Proof.** We prove the theorem in three parts.

**Anamorphic security.** We use a sequence of hybrid games $\mathsf{Game}\ 0, \mathsf{Game}\ (1, j)$ for $j \in [0, Q]$ to prove this theorem. In each hybrid game, we assume $\mathcal{D}$ is a PPT dictator and $Q$ is the total number of queries made by $\mathcal{D}$ to the encryption oracle. Let $Q_{\mathsf{H}_1}$ and $Q_{\mathsf{H}_2}$ denote the number of hash queries made by $\mathcal{D}$.

Game 0**:** This is the anamorphic $\mathsf{GAnam}_{2\mathsf{BF}}^{\mathcal{D}}(\lambda)$. The challenger generates $(\mathsf{dmsk} = \alpha, \mathsf{dmpk} = P^\alpha) \leftarrow \mathsf{aBF.aSetup}(1^\lambda)$, and sends the key pair $(\mathsf{ampk}, \mathsf{amsk})$ to $\mathcal{D}$. Then, $\mathcal{D}$ makes polynomial number of queries to the encryption oracle with a tuple $\{(\mathsf{id}_i, \mathbf{msg}_i, \mathsf{amsg}_i)\}_{i \in [Q]}$. At the $i$-th query, $\mathcal{D}$ receives a ciphertext $\mathbf{act}_i$ computed as follows:

1. Parse $\mathbf{msg}_i = (\mathsf{msg}_{i_1}, \mathsf{msg}_{i_2})$
2. $Q_{\mathsf{id}_i} \coloneqq \mathsf{H}_1(\mathsf{id}_i)$
3. $\widehat{g}_{\mathsf{id}_i} \coloneqq e(Q_{\mathsf{id}_i}, \mathsf{dmpk})$, $g_{\mathsf{id}_i} \coloneqq e(Q_{\mathsf{id}_i}, P_{\mathsf{pub}})$
4. $\kappa_1, \kappa_2 \leftarrow \mathbb{Z}_q^*$ s.t. $\mathsf{H}_2(g_{\mathsf{id}_i}^{\kappa_2}) = \mathsf{amsg}_i \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}_i}^{\kappa_1})$
5. $\mathsf{act}_1 \coloneqq \mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_{i_1}, \mathsf{id}_i; \kappa_1)$
6. $\mathsf{act}_2 \coloneqq \mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_{i_2}, \mathsf{id}_i; \kappa_2)$
7. **return** $\mathbf{act}_i \coloneqq (\mathsf{act}_{i_1}, \mathsf{act}_{i_2})$

Game $(1, j)$ **for** $j \in [0, Q]$**:** We define $\mathsf{Game}\ (1, 0)$ to be identical to $\mathsf{Game}\ 0$, and $\mathsf{Game}\ (1, j)$ is exactly the same as $\mathsf{Game}\ (1, 0)$ except that the first $j$ (anamorphic) ciphertexts $\{\mathbf{act}_i\}_{i \in [1, j]}$ are computed as follows:

1. Parse $\mathbf{msg}_i = (\mathsf{msg}_{i_1}, \mathsf{msg}_{i_2})$
2. $\kappa_1, \kappa_2 \leftarrow \mathbb{Z}_q^*$.
3. $\mathsf{act}_{i_1} \coloneqq \mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_{i_1}, \mathsf{id}_i; \kappa_1) = (r_1 = P^{\kappa_1}, c_1 = \mathsf{msg}_{i_1} \oplus \mathsf{H}_2(g_{\mathsf{id}_i}^{\kappa_1}))$
4. $\mathsf{act}_{i_2} \coloneqq \mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_{i_2}, \mathsf{id}_i; \kappa_2) = (r_2 = P^{\kappa_2}, c_2 = \mathsf{msg}_{i_2} \oplus \mathsf{H}_2(g_{\mathsf{id}_i}^{\kappa_2}))$
5. **return** $\mathbf{act}_i \coloneqq (\mathsf{act}_{i_1}, \mathsf{act}_{i_2})$

We show that $\mathsf{Game}\ (1,j)$ is indistinguishable from $\mathsf{Game}\ (1,j-1)$ by the DBDH assumption. The only difference between $\mathsf{Game}\ (1,j)$ and $\mathsf{Game}\ (1,j-1)$ is in the computation of $\kappa_1,\kappa_2$. In particular, $\kappa_1,\kappa_2$ are sampled from $\mathbb{Z}_q^*$ such that $\mathsf{H}_2(g_{\mathsf{id}_i}^{\kappa_2}) = \mathsf{amsg}_i \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}_i}^{\kappa_1})$ in $\mathsf{Game}\ (1,j-1)$, whereas $\kappa_1,\kappa_2$ are sampled uniformly random from $\mathbb{Z}_q^*$ in $\mathsf{Game}\ (1,j)$. We first show that $\widehat{g}_{\mathsf{id}_i}^{\kappa_1}$ is indistinguishable from a uniformly random element in $\mathbb{G}_T$ in $\mathcal{D}$'s view. Since $\mathsf{H}_1$ is modeled as ROM, we can set $Q_{\mathsf{id}_i} = P^r$ for some $r \leftarrow \mathbb{Z}_q$. More precisely, the simulator answers $\mathsf{H}_1$-queries lazily; For the $j$-th oracle call, the simulator embeds $P^r$ at $\mathsf{H}_1(\mathsf{id}_j)$ using a standard guessing strategy. At the start, the simulator guesses an index $i^* \in [Q_{\mathsf{H}_1}]$ uniformly at random, and embeds $P^r$ in the reply to the $i^*$-th $\mathsf{H}_1$ query. With probability $1/Q_{\mathsf{H}_1}$, the guess matches the $\mathsf{id}_j$ queried by $\mathcal{D}$. If at the time of $j$-th oracle call, the identity $\mathsf{id}_j$ chosen by $\mathcal{D}$ does not equal $\mathsf{id}_{i^*}$, the simulator aborts and outputs a uniformly random bit. This incurs the usual polynomial $1/Q_{\mathsf{H}_1}$ factor in the reduction. The remaining $\mathsf{H}_1$ queries are answered with freshly sampled group elements.

Without loss of generality, we assume that $\kappa_1$ is first sampled uniformly at random from $\mathbb{Z}_q$ and then $\kappa_2$ is picked from $\mathbb{Z}_q^*$ such that $\mathsf{H}_2(g_{\mathsf{id}_i}^{\kappa_2}) = \mathsf{amsg}_i \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}_i}^{\kappa_1})$ holds. Thus the challenger can simulate $\mathsf{Game}\ (1,j)$ with $(P^r, P^\alpha, P^{\kappa_1}, e(P,P)^t)$ when $t = r\alpha\kappa_1$, and $\mathsf{Game}\ (1,j)$, when $t$ is uniformly random in $\mathbb{Z}_q$, where it sets $\widehat{g}_{\mathsf{id}_i}^{\kappa_1} = e(P,P)^t$. Therefore, by the DBDH assumption the following holds:

$$\begin{aligned}|\Pr[r,\alpha,\kappa_1 \leftarrow \mathbb{Z}_q : \mathcal{A}(P^r, P^\alpha, P^{\kappa_1}, e(P,P)^{r\cdot\alpha\cdot\kappa_1}) = 1] - \\ \Pr[r,\alpha,\kappa_1,t \leftarrow \mathbb{Z}_q : \mathcal{A}(P^r, P^\alpha, P^{\kappa_1}, e(P,P)^t) = 1]| \le \mathsf{negl}(\lambda)\end{aligned}$$

Since $t = r\cdot\alpha\cdot\kappa_1$ is indistinguishable from uniformly random and $\mathsf{H}_2$ is modeled as ROM, $\mathsf{H}_2(\widehat{g}_{\mathsf{id}_i}^{\kappa_1}) \oplus \mathsf{amsg}_i$ is statistically close to $\mathsf{H}_2(\widehat{g}_{\mathsf{id}_i}^{\kappa_1})$. More specifically, the simulator programs $\mathsf{H}_2(\widehat{g}_{\mathsf{id}_i}^{\kappa_1})$ in each oracle call. Since $\widehat{g}_{\mathsf{id}_i}^{\kappa_1}$ is (computationally) uniform in $\mathbb{G}_T$ under DBDH and independent of $\mathcal{D}$'s view until the $i$-th oracle call releases $\mathsf{act}_{i_1}, \mathsf{act}_{i_2}$, the probability that $\mathcal{D}$ previously queried $\mathsf{H}_2$ at this point is at most $Q_{\mathsf{H}_2}/q$ (plus a DBDH negligible term), and the total abort probability over all $Q$ oracle calls is at most $Q\cdot Q_{\mathsf{H}_2}/q$.

Furthermore, at this stage, $g_{\mathsf{id}_i}^{\kappa_2}$ and $\widehat{g}_{\mathsf{id}_i}^{\kappa_1}$ are independently distributed where $\kappa_1,\kappa_2 \leftarrow \mathbb{Z}_q^*$ and hence the hybrids $\mathsf{Game}\ (1,j)$ and $\mathsf{Game}\ (1,j-1)$ are indistinguishable.
Finally, we observe that $\mathsf{Game}\ (1,Q)$ is exactly the same as $\mathsf{GReal}_{\mathsf{2BF}}^{\mathcal{D}}(\lambda)$. This completes the proof.

**Fully asymmetric IND-CPA security.** We prove the security using hybrid games $\mathsf{Game}\ 0, \ldots, \mathsf{Game}\ 3$. In each hybrid, we assume $\mathcal{A}$ is a PPT adversary.

Game 0**:** This is the IND-CPA security experiment $\mathsf{GIND\text{-}CPA}_{\mathsf{AAE}}^{\mathcal{A}}(\lambda,0)$ defined as follows:

1: $(\mathsf{mpk} \coloneqq (\mathbb{G},\mathbb{G}_T,P,e,q,P_{\mathsf{pub}},\mathsf{H}_1,\mathsf{H}_2),\ \mathsf{msk} \coloneqq s) \leftarrow \mathsf{BF.Setup}(1^\lambda)$
2: $\alpha \leftarrow \mathbb{Z}_q^*$
3: $\mathsf{ampk} \coloneqq \mathsf{mpk}, \mathsf{amsk} \coloneqq \mathsf{msk}$
4: $\mathsf{dmpk} \coloneqq P^\alpha, \mathsf{dmsk} \coloneqq \alpha, \mathcal{S} = \emptyset, \mathcal{K} = \emptyset$
5: Key queries: $\mathcal{A}(\mathsf{ampk},\mathsf{dmpk})$ sends $\mathsf{id}$
$\mathcal{A}$ gets $(\mathsf{H}_1(\mathsf{id})^{\mathsf{amsk}}, \mathsf{H}_1(\mathsf{id})^{\mathsf{dmsk}})$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathsf{id}\},\ \mathcal{K} \leftarrow \mathcal{K} \cup \{(\mathsf{H}_1(\mathsf{id})^{\mathsf{amsk}}, \mathsf{H}_1(\mathsf{id})^{\mathsf{dmsk}})\}$
6: $(\mathsf{id}^*, \mathbf{msg}_0, \mathbf{msg}_1, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{ampk},\mathsf{dmpk},\mathcal{S},\mathcal{K})$
7: Parse $\mathbf{msg}_0 = (\mathsf{msg}_{0,1}, \mathsf{msg}_{0,2})$
8: Parse $\mathbf{msg}_1 = (\mathsf{msg}_{1,1}, \mathsf{msg}_{1,2})$
9: $Q_{\mathsf{id}^*} \coloneqq \mathsf{H}_1(\mathsf{id}^*)$
10: $\widehat{g}_{\mathsf{id}^*} \coloneqq e(Q_{\mathsf{id}^*}, \mathsf{dmpk}),\ g_{\mathsf{id}^*} \coloneqq e(Q_{\mathsf{id}^*}, P_{\mathsf{pub}})$
11: $\kappa_1,\kappa_2 \leftarrow \mathbb{Z}_q^*$ s.t. $\mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_2}) = \mathsf{amsg}_0 \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}^*}^{\kappa_1})$
12: $\mathsf{act}_1 = (r_1,c_1) \coloneqq \mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_{0,1}, \mathsf{id}; \kappa_1)$
// $r_1 = P^{\kappa_1}, c_1 = \mathsf{msg}_{0,1} \oplus \mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_1})$

13: $\mathsf{act}_2 = (r_2, c_2) \coloneqq \mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_{0,2}, \mathsf{id}; \kappa_2)$

// $r_2 = P^{\kappa_2}, c_2 = \mathsf{msg}_{0,2} \oplus \mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_2})$

14: Repeat step 5

15: **return** $\beta' \leftarrow \mathcal{A}(\mathbf{act}^* = (\mathsf{act}_1^*, \mathsf{act}_2^*))$ if $\mathsf{id}^* \notin \mathcal{S}$

Game 1: This is the same as Game 0 except that the challenger replaces $\mathsf{amsg}_0 \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}^*}^{\kappa_1})$ with a uniformly random bit $c^*$. In particular, the challenger samples $\kappa_2 \leftarrow \mathbb{Z}_q^*, c^* \leftarrow \{0,1\}$ and sets $\mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_2}) = c^*$. The indistinguishability between Game 0 and Game 1 follows from the IND$-CPA security of the Boneh-Franklin BasicIdent scheme[8] under the public key dmpk.

We show that if there is an adversary $\mathcal{A}$ that breaks the indistinguishability between the two games, then there exists another adversary $\mathcal{B}$ which breaks the IND$-CPA security of the Boneh-Franklin BasicIdent scheme. More specifically, $\mathcal{B}$, having received $\mathsf{mpk} = (\mathbb{G}, \mathbb{G}_T, P, e, q, P^\alpha, \mathsf{H}_1, \mathsf{H}_2)$ from its challenger, simulates the adversary $\mathcal{A}$ as follows:

Simulator $\mathcal{B}(1^\lambda, \mathsf{mpk})$

1: $s \leftarrow \mathbb{Z}_q^*$

2: $\mathsf{ampk} \coloneqq (\mathbb{G}, \mathbb{G}_T, P, e, q, P_{\mathsf{pub}} = P^s, \mathsf{H}_1, \mathsf{H}_2)$, $\mathsf{amsk} \coloneqq s$

3: $\mathsf{dmpk} \coloneqq P^\alpha, \mathcal{S} = \emptyset, \mathcal{K} = \emptyset$

4: Key queries: $\mathcal{A}(\mathsf{ampk}, \mathsf{dmpk})$ sends $\mathsf{id}$

$\mathcal{B}$ sends $\mathsf{id}$ to its challenger and gets $\mathsf{dsk}_{\mathsf{id}}$

$\mathcal{A}$ gets $(\mathsf{H}_1(\mathsf{id})^{\mathsf{amsk}}, \mathsf{dsk}_{\mathsf{id}})$

$\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathsf{id}\},\ \mathcal{K} \leftarrow \mathcal{K} \cup \{(\mathsf{H}_1(\mathsf{id})^{\mathsf{amsk}}, \mathsf{dsk}_{\mathsf{id}})\}$

5: $(\mathsf{id}^*, \mathbf{msg}_0, \mathbf{msg}_1, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{ampk}, \mathsf{dmpk}, \mathcal{S}, \mathcal{K})$

6: Parse $\mathbf{msg}_0 = (\mathsf{msg}_{0,1}, \mathsf{msg}_{0,2})$

7: Parse $\mathbf{msg}_1 = (\mathsf{msg}_{1,1}, \mathsf{msg}_{1,2})$

8: $\mathcal{B}$ sends $(\mathsf{id}^*, \mathsf{amsg}_0, \mathsf{amsg}_1)$ to its challenger and gets $(r^*, c^*)$

9: $Q_{\mathsf{id}^*} \coloneqq \mathsf{H}_1(\mathsf{id}^*)$

10: $g_{\mathsf{id}^*} \coloneqq e(Q_{\mathsf{id}^*}, P_{\mathsf{pub}}) = e(Q_{\mathsf{id}^*}, P)^{\mathsf{amsk}}$

11: $\kappa_2 \leftarrow \mathbb{Z}_q^*$ s.t. $\mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_2}) = c^*$

12: $\mathsf{act}_1 \coloneqq (r_1, c_1)$ where $r_1 \coloneqq r^*, c_1 \coloneqq \mathsf{msg}_{0,1} \oplus \mathsf{H}_2(e(Q_{\mathsf{id}^*}, r^*)^{\mathsf{amsk}})$

13: $\mathsf{act}_2 = (r_2, c_2) \coloneqq \mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_{0,2}, \mathsf{id}; \kappa_2)$

// $r_2 = P^{\kappa_2}, c_2 = \mathsf{msg}_{0,2} \oplus \mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_2})$

14: Repeat step 5

15: $\mathcal{B}$ simulates all the hash queries of $\mathcal{A}$ using its own challenger

16: **return** $\beta' \leftarrow \mathcal{A}(\mathbf{act}^* = (\mathsf{act}_1^*, \mathsf{act}_2^*))$ if $\mathsf{id}^* \notin \mathcal{S}$

It can be observed that $\mathcal{B}$ perfectly simulates $\mathcal{A}$. Firstly, if $(r^*, c^*)$ takes the form $(r^* = P^{\kappa_1}, c^* = \mathsf{amsg}_0 \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}^*}^{\kappa_1}))$, i.e. the BasicIdent scheme encrypts $\mathsf{amsg}_0$ using mpk (or dmpk in the view of $\mathcal{A}$), then $\mathcal{B}$ simulates Game 0 for $\mathcal{A}$. The only difference is in step 12, where $\mathsf{act}_1$ is computed as $\mathsf{act}_1 = (r_1, c_1)$. However, we observe that

$$r_1 = r^* = P^{\kappa_1}, c_1 = \mathsf{msg}_{0,1} \oplus \mathsf{H}_2(e(Q_{\mathsf{id}^*}, r^*)^{\mathsf{amsk}}) = \mathsf{msg}_{0,1} \oplus \mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_1})$$

[8] It is easy to see that Boneh-Franklin BasicIdent scheme satisfies the IND$-CPA security under the BDH problem [BF01], which ensures that the challenge ciphertext can be sampled uniformly at random from the ciphertext space $\mathbb{G} \times \{0,1\}$.

which is a valid ciphertext encrypting $\mathsf{msg}_{0,1}$ under $\mathsf{ampk}$. On the other hand, if the $\mathsf{BasicIdent}$ scheme picks a random ciphertext $(r^*, c^*)$ in the step 8, then $\mathcal{B}$ samples $\kappa_2 \leftarrow \mathbb{Z}_q^*$ such that $\mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_2}) = c^*$ in step 11. In this case, $\mathcal{B}$ simulates $\mathsf{Game}$ 1 for $\mathcal{A}$. Hence, the indistinguishability between $\mathsf{Game}$ 0 and $\mathsf{Game}$ 1 follows from the IND$-CPA security of the Boneh-Franklin $\mathsf{BasicIdent}$ scheme under the public key $\mathsf{mpk}$.

Game 2: This is the same as $\mathsf{Game}$ 1 except that the challenger encrypts $\mathbf{msg}_1$ instead of $\mathbf{msg}_0$:

1: $(\mathsf{mpk} \coloneqq (\mathbb{G}, \mathbb{G}_T, P, e, q, P_{\mathsf{pub}}, \mathsf{H}_1, \mathsf{H}_2),\ \mathsf{msk} \coloneqq s) \leftarrow \mathsf{BF.Setup}(1^\lambda)$

2: $\alpha \leftarrow \mathbb{Z}_q^*$

3: $\mathsf{ampk} \coloneqq \mathsf{mpk}, \mathsf{amsk} \coloneqq \mathsf{msk}$

4: $\mathsf{dmpk} \coloneqq P^\alpha, \mathsf{dmsk} \coloneqq \alpha, \mathcal{S} = \emptyset, \mathcal{K} = \emptyset$

5: Key queries: $\mathcal{A}(\mathsf{ampk}, \mathsf{dmpk})$ sends $\mathsf{id}$

$\mathcal{A}$ gets $(\mathsf{H}_1(\mathsf{id})^{\mathsf{amsk}}, \mathsf{H}_1(\mathsf{id})^{\mathsf{dmsk}})$

$\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathsf{id}\},\ \mathcal{K} \leftarrow \mathcal{K} \cup \{(\mathsf{H}_1(\mathsf{id})^{\mathsf{amsk}}, \mathsf{H}_1(\mathsf{id})^{\mathsf{dmsk}})\}$

6: $(\mathsf{id}^*, \mathbf{msg}_0, \mathbf{msg}_1, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{ampk}, \mathsf{dmpk}, \mathcal{S}, \mathcal{K})$

7: Parse $\mathbf{msg}_0 = (\mathsf{msg}_{0,1}, \mathsf{msg}_{0,2})$

8: Parse $\mathbf{msg}_1 = (\mathsf{msg}_{1,1}, \mathsf{msg}_{1,2})$

9: $Q_{\mathsf{id}^*} \coloneqq \mathsf{H}_1(\mathsf{id}^*)$

10: $\widehat{g}_{\mathsf{id}^*} \coloneqq e(Q_{\mathsf{id}^*}, \mathsf{dmpk}),\ g_{\mathsf{id}^*} \coloneqq e(Q_{\mathsf{id}^*}, P_{\mathsf{pub}})$

11: $\kappa_1, \kappa_2 \leftarrow \mathbb{Z}_q^*, c^* \leftarrow \{0,1\}$ s.t. $\mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_2}) = c^*$

12: $\mathsf{act}_1 = (r_1, c_1) \coloneqq \mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_{1,1}, \mathsf{id}; \kappa_1)$

// $r_1 = P^{\kappa_1}, c_1 = \mathsf{msg}_{1,1} \oplus \mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_1})$

13: $\mathsf{act}_2 = (r_2, c_2) \coloneqq \mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_{1,2}, \mathsf{id}; \kappa_2)$

// $r_2 = P^{\kappa_2}, c_2 = \mathsf{msg}_{1,2} \oplus \mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_2})$

14: Repeat step 5

15: **return** $\beta' \leftarrow \mathcal{A}(\mathbf{act}^* = (\mathsf{act}_1^*, \mathsf{act}_2^*))$ if $\mathsf{id}^* \notin \mathcal{S}$

The indistinguishability between $\mathsf{Game}$ 1 and $\mathsf{Game}$ 2 follows the IND-CPA security of the Boneh-Franklin $\mathsf{BasicIndent}$ scheme, in particular from the $\mathsf{BDH}$ assumption.

Game 3: This is the same as $\mathsf{Game}$ 2 except that the challenger sets $c^* = \mathsf{amsg}_1 \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}^*}^{\kappa_1})$ instead of sampling it uniformly at random from $\{0,1\}$:

1: $(\mathsf{mpk} \coloneqq (\mathbb{G}, \mathbb{G}_T, P, e, q, P_{\mathsf{pub}}, \mathsf{H}_1, \mathsf{H}_2),\ \mathsf{msk} \coloneqq s) \leftarrow \mathsf{BF.Setup}(1^\lambda)$

2: $\alpha \leftarrow \mathbb{Z}_q^*$

3: $\mathsf{ampk} \coloneqq \mathsf{mpk}, \mathsf{amsk} \coloneqq \mathsf{msk}$

4: $\mathsf{dmpk} \coloneqq P^\alpha, \mathsf{dmsk} \coloneqq \alpha, \mathcal{S} = \emptyset, \mathcal{K} = \emptyset$

5: Key queries: $\mathcal{A}(\mathsf{ampk}, \mathsf{dmpk})$ sends $\mathsf{id}$

$\mathcal{A}$ gets $(\mathsf{H}_1(\mathsf{id})^{\mathsf{amsk}}, \mathsf{H}_1(\mathsf{id})^{\mathsf{dmsk}})$

$\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathsf{id}\},\ \mathcal{K} \leftarrow \mathcal{K} \cup \{(\mathsf{H}_1(\mathsf{id})^{\mathsf{amsk}}, \mathsf{H}_1(\mathsf{id})^{\mathsf{dmsk}})\}$

6: $(\mathsf{id}^*, \mathbf{msg}_0, \mathbf{msg}_1, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{ampk}, \mathsf{dmpk}, \mathcal{S}, \mathcal{K})$

7: Parse $\mathbf{msg}_0 = (\mathsf{msg}_{0,1}, \mathsf{msg}_{0,2})$

8: Parse $\mathbf{msg}_1 = (\mathsf{msg}_{1,1}, \mathsf{msg}_{1,2})$

9: $Q_{\mathsf{id}^*} \coloneqq \mathsf{H}_1(\mathsf{id}^*)$

10: $\widehat{g}_{\mathsf{id}^*} \coloneqq e(Q_{\mathsf{id}^*}, \mathsf{dmpk})$, $g_{\mathsf{id}^*} \coloneqq e(Q_{\mathsf{id}^*}, P_{\mathsf{pub}})$

11: $\kappa_1, \kappa_2 \leftarrow \mathbb{Z}_q^*$ s.t. $\mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_2}) = \mathsf{amsg}_1 \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}^*}^{\kappa_1})$

12: $\mathsf{act}_1 = (r_1, c_1) \coloneqq \mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_{1,1}, \mathsf{id}; \kappa_1)$

// $r_1 = P^{\kappa_1}, c_1 = \mathsf{msg}_{1,1} \oplus \mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_1})$

13: $\mathsf{act}_2 = (r_2, c_2) \coloneqq \mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_{1,2}, \mathsf{id}; \kappa_2)$

// $r_2 = P^{\kappa_2}, c_2 = \mathsf{msg}_{1,2} \oplus \mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_2})$

14: Repeat step 5

15: **return** $\beta' \leftarrow \mathcal{A}(\mathbf{act}^* = (\mathsf{act}_1^*, \mathsf{act}_2^*))$ if $\mathsf{id}^* \notin \mathcal{S}$

The indistinguishability follows the IND$-CPA security of the Boneh-Franklin BasicIdent scheme under the public key dmpk similar to Game 1.

Finally, we observe that Game 3 is exactly the same as $\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBAE}}(\lambda, 1)$. This concludes the fully asymmetric security of $\Pi_{\mathsf{aBF}}$.

**sIND-CPA security.** We prove the strong IND-CPA security by designing a simulator that plays the role of the challenger in the IND-CPA security game of the Boneh-Franklin BasicIdent scheme against ampk, which is IND-CPA secure under the BDH problem. Note that the anamorphic master public key ampk and the master public key of the BasicIdent scheme are identically distributed since the Boneh-Franklin BasicIdent scheme is IBAE-compatible by Lemma 4. In other words, if an adversary breaks the sIND-CPA security of $\Pi_{\mathsf{IBAE}}$ then it can be used to break the IND-CPA security of Boneh-Franklin BasicIdent scheme.

We start with a contradiction that there exists a PPT adversary $\mathcal{A}$ that breaks the sIND-CPA security of $\Pi_{\mathsf{aBF}}$ with a non-negligible probability. We then design another adversary $\mathcal{B}$ that breaks IND-CPA security of the BasicIdent scheme. The challenger of $\mathcal{B}$ runs the setup of the BasicIndent scheme and returns $\mathsf{mpk} = (\mathbb{G}, \mathbb{G}_T, P, e, q, P^{\alpha}, \mathsf{H}_1, \mathsf{H}_2)$ to $\mathcal{B}$. Then, $\mathcal{B}$ utilizes $\mathcal{A}$ to break the IND-CPA security of the BasicIndent scheme as follows:

Simulator $\mathcal{B}(1^{\lambda}, \mathsf{mpk})$

1: $s \leftarrow \mathbb{Z}_q^*$

2: $\mathsf{ampk} \coloneqq (\mathbb{G}, \mathbb{G}_T, P, e, q, P_{\mathsf{pub}} = P^s, \mathsf{H}_1, \mathsf{H}_2)$, $\mathsf{amsk} \coloneqq s$

3: $\mathsf{dmpk} \coloneqq P^{\alpha}, \mathcal{S} = \emptyset, \mathcal{K} = \emptyset$

4: Key queries: $\mathcal{A}(\mathsf{ampk}, \mathsf{amsk}, \mathsf{dmpk})$ sends $\mathsf{id}$

$\mathcal{B}$ sends $\mathsf{id}$ to its challenger and gets $\mathsf{dsk}_{\mathsf{id}}$

$\mathcal{A}$ gets $(\mathsf{H}_1(\mathsf{id})^{\mathsf{amsk}}, \mathsf{dsk}_{\mathsf{id}})$

$\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathsf{id}\}$, $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\mathsf{H}_1(\mathsf{id})^{\mathsf{amsk}}, \mathsf{dsk}_{\mathsf{id}})\}$

5: $(\mathsf{id}^*, \mathbf{msg}, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{ampk}, \mathsf{amsk}, \mathsf{dmpk}, \mathcal{S}, \mathcal{K})$

6: Parse $\mathbf{msg} = (\mathsf{msg}_1, \mathsf{msg}_2)$

7: $\mathcal{B}$ sends $(\mathsf{id}^*, \mathsf{amsg}_0, \mathsf{amsg}_1)$ to its challenger and gets $(r^*, c^*)$

8: $Q_{\mathsf{id}^*} \coloneqq \mathsf{H}_1(\mathsf{id}^*)$

9: $g_{\mathsf{id}^*} \coloneqq e(Q_{\mathsf{id}^*}, P_{\mathsf{pub}}) = e(Q_{\mathsf{id}^*}, P)^{\mathsf{amsk}}$

10: $\kappa_2 \leftarrow \mathbb{Z}_q^*$ s.t. $\mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_2}) = c^*$

11: $\mathsf{act}_1 \coloneqq (r_1, c_1)$ where $r_1 \coloneqq r^*, c_1 \coloneqq \mathsf{msg}_1 \oplus \mathsf{H}_2(e(Q_{\mathsf{id}^*}, r^*)^{\mathsf{amsk}})$

12: $\mathsf{act}_2 \coloneqq (r_2, c_2) \coloneqq \mathsf{BF.Enc}(\mathsf{ampk}, \mathsf{msg}_2, \mathsf{id}; \kappa_2)$

// $r_2 = P^{\kappa_2}, c_2 = \mathsf{msg}_2 \oplus \mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_2})$

13: Repeat step 5

14: $\mathcal{B}$ simulates all the hash queries of $\mathcal{A}$ using its own challenger

15: **return** $\beta' \leftarrow \mathcal{A}(\mathbf{act}^* = (\mathsf{act}_1^*, \mathsf{act}_2^*))$ if $\mathsf{id}^* \notin \mathcal{S}$

It can be observed that if $(r^*, c^*)$ takes the form $(r^* = P^{\kappa_1}, c^* = \mathsf{amsg}_\beta \oplus \mathsf{H}_2(\widehat{g}_{\mathsf{id}^*}^{\kappa_1}))$, i.e. the $\mathsf{BasicIdent}$ scheme encrypts $\mathsf{amsg}_\beta$ using $\mathsf{mpk}$ (or $\mathsf{dmpk}$ in the view of $\mathcal{A}$), then $\mathcal{B}$ perfectly simulates $\mathsf{GsIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{IBAE}}(\lambda, \beta)$ for $\mathcal{A}$. The only difference is in step 11, where $\mathsf{act}_1$ is computed as $\mathsf{act}_1 = (r_1, c_1)$. However, we observe that

$$r_1 = r^* = P^{\kappa_1}, c_1 = \mathsf{msg}_1 \oplus \mathsf{H}_2(e(Q_{\mathsf{id}^*}, r^*)^{\mathsf{amsk}}) = \mathsf{msg}_1 \oplus \mathsf{H}_2(g_{\mathsf{id}^*}^{\kappa_1})$$

which is a valid $\mathsf{BF}$ ciphertext encrypting $\mathsf{msg}_1$ under $\mathsf{ampk}$. Hence, if $\mathcal{A}$ breaks the sIND-CPA security of $\Pi_{\mathsf{aBF}}$ with a non-negligible probability, then $\mathcal{B}$ can also break the IND-CPA security of the Boneh-Franklin $\mathsf{BasicIdent}$ scheme under the public key $\mathsf{mpk}$. This completes the proof.

□

## 5.2 Extension of FO-Based IBE

Variants of the FO transformation are also a viable tool to construct CCA secure IBE schemes. This has started with the intial construction due to Boneh and Franklin on IBE [BF01] and several works have explicitly investigated various FO-type transformations for lifting OW-CPA or IND-CPA secure IBE to CCA secure ones [Gal05, YKH+06, KYH+06]. Analogous to the discussion of the FO transformation in Section 4.3, it is natural to ask whether we can equivalently apply these results to the IBAE setting. We first want to briefly recall how the variant of the FO transform that is used in the Boneh-Franklin (BF) IBE and the Sakai-Kasahara (SK) IBE [SOK01, CC05] exactly work. We note that both works as specified below are suggested in an ETSI technical report for standardization [ETS22][9].

For our presentation let $\Pi_{\mathsf{IBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ and $\Pi_{\mathsf{SKE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be the underlying IBE and symmetric encryption schemes. Analogous to Section 4.3 we now denote by $\Pi_{\mathsf{FO,IBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ the IND-CCA secure version where $\mathsf{FO}_{\mathsf{IBE}}.\mathsf{Setup}$ and $\mathsf{FO}_{\mathsf{IBE}}.\mathsf{KeyGen}$ just run $\mathsf{IBE.Setup}$ and $\mathsf{IBE.KeyGen}$ respectively. $\mathsf{FO}_{\mathsf{IBE}}.\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{msg})$ returns ciphertext $c = (c_1, c_2)$ where $c_1 = \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{msg}; \sigma)$ and $c_2 = \mathsf{SKE.Enc}(G(r), \mathsf{msg})$ for uniformly random $r \in \{0,1\}^n$ (and suitable $n$) and $\sigma = H(r, \mathsf{msg}) \in \mathcal{M}_{\mathsf{IBE}}$. Note that $H$ and $G$ are hash functions modeled as random oracles mapping to the space of random coins $\mathcal{R}_{\mathsf{IBE}}$ of $\mathsf{IBE}$ and the key space $\mathcal{K}_{\mathsf{SKE}}$ of $\mathsf{SKE}$, respectively. For decryption, $\mathsf{FO}_{\mathsf{IBE}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, c)$ parses $c = (c_1, c_2)$ computes $r' = \mathsf{IBE.Dec}(\mathsf{sk}_{\mathsf{id}}, c_1)$ and $\mathsf{msg}' = \mathsf{SKE.Dec}(G(r'), c_2)$ and returns $\mathsf{msg}'$ if $c_1' = \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{id}, H(r', \mathsf{msg}'))$ or $\perp$ otherwise.

We present the anamorphic IBE extension based on $\Pi_{\mathsf{FO,IBE}}$ in Figure 6.

**Theorem 6** *Let $\Pi_{\mathsf{FO,IBE}}$ be an $\mathsf{IND\text{-}ID\text{-}CCA}$ secure FO-based IBE where the underlying IBE $\Pi_{\mathsf{IBE}}$ is IND$-CPA secure and has an exponentially large message space $\mathcal{M}_{\mathsf{IBE}}$ (i.e., $|\mathcal{M}_{\mathsf{IBE}}| \geq 2^{\mathsf{poly}(\lambda)}$ for non-constant $\mathsf{poly}$), and the underlying SKE $\Pi_{\mathsf{SKE}}$ is wIND$-CPA secure. Furthermore, let $(\mathsf{Encode}, \mathsf{Decode})$ be a pseudo-random $(\mathcal{C}_{\mathsf{FO}_{\mathsf{IBE}}}, \mathcal{M}_{\mathsf{IBE}}^{\mu})$-encoding, where $\mathcal{C}_{\mathsf{FO}_{\mathsf{IBE}}}$ denotes the ciphertext space of $\Pi_{\mathsf{FO,IBE}}$. Then $\Pi_{\mathsf{FOAE,IBE}}$ as defined in Figure 6 is a $\mu$-message anamorphic identity-based encryption extension satisfying anamorphic security and sIND-RCCA security in the ROM.*

[9] The only difference with the standard, as with the papers, is that they explicitly consider XOR as symmetric encryption scheme $\mathsf{SKE}$.

$\underline{\mathsf{aSetup}(1^\lambda)}$ :

1. $(\mathsf{amsk}, \mathsf{ampk}) \leftarrow \mathsf{FO}_{\mathsf{IBE}}.\mathsf{Setup}(1^\lambda)$
2. $(\mathsf{dmsk}, \mathsf{dmpk}) \leftarrow \mathsf{FO}_{\mathsf{IBE}}.\mathsf{Setup}(1^\lambda)$
3. **return** $(\mathsf{amsk}, \mathsf{ampk}, \mathsf{dmpk}, \mathsf{dmsk})$

$\underline{\mathsf{aDec}(\mathsf{ask}_{\mathsf{id}}, \mathsf{dsk}_{\mathsf{id}}, \mathbf{act})}$ :

1. **for** $j \in \{1, \ldots, \mu\}$:
   Parse $\mathsf{act}_j = (c_1, c_2)$
   $r_j := \mathsf{IBE}.\mathsf{Dec}(\mathsf{ask}_{\mathsf{id}}, c_1)$
   **if** $r_j = \perp$ **return** $\perp$
2. $c := \mathsf{Decode}(r_1, \ldots, r_\mu)$
3. **if** $c = \perp$ **return** $\perp$
4. **return** $\mathsf{FO}_{\mathsf{IBE}}.\mathsf{Dec}(\mathsf{dsk}_{\mathsf{id}}, c)$

$\underline{\mathsf{aKeyGen}(\mathsf{dmsk}, \mathsf{id})}$ :

1. $\mathsf{dsk}_{\mathsf{id}} = \mathsf{FO}_{\mathsf{IBE}}.\mathsf{KeyGen}(\mathsf{dmsk}, \mathsf{id})$
2. **return** $\mathsf{dsk}_{\mathsf{id}}$

$\underline{\mathsf{aEnc}(\mathsf{ampk}, \mathsf{dmpk}, \mathsf{id}, \mathbf{msg}, \mathsf{amsg})}$ :

1. Parse $\mathbf{msg} = (\mathsf{msg}_1, \ldots, \mathsf{msg}_\mu)$
2. $c \leftarrow \mathsf{FO}_{\mathsf{IBE}}.\mathsf{Enc}(\mathsf{dmpk}, \mathsf{id}, \mathsf{amsg})$
3. $(r_1, \ldots, r_\mu) \leftarrow \mathsf{Encode}(c)$
4. **for** $j \in \{1, \ldots, \mu\}$:
   $\mathsf{act}_j := \mathsf{FO}_{\mathsf{IBE}}.\mathsf{Enc}(\mathsf{ampk}, \mathsf{id}, \mathsf{msg}_j; r_j)$
5. **return act** $:= (\mathsf{act}_1, \ldots, \mathsf{act}_\mu)$

Figure 6: $\mu$-AIBE $\Pi_{\mathsf{FOAE,IBE}}$

The proof of this theorem works analogous to that of Theorem 3 and is thus omitted. The intuition is that $\Pi_{\mathsf{FO,IBE}}$ satisfies anamorphic security as the embedded ciphertexts components containing $\mathsf{amsg}$ are indistinguishable from a regular $r \leftarrow \mathcal{M}_{\mathsf{IBE}}$ due to the pseud-randomness of the pseudo-random encoding. Moreover, the IND-CCA security of $\Pi_{\mathsf{FO,IBE}}$ for encrypting $\mathsf{amsg}$ (i.e., the "inner" layer encryption) implies sIND-RCCA for the overall construction $\Pi_{\mathsf{FOAE,IBE}}$ as an adversary (including the dictator) is not able to maul $\mathsf{amsg}$.

**Instantiation with BF Full-Indent.** Let us consider $\Pi_{\mathsf{FO,IBE}}$ based on Boneh-Frankin (BF) as IBE and XOR for the SKE part. BF satisfies IND\$-CPA security under DBDH in the ROM and XORing a message with a random key of the same length satisfies wIND\$-CPA security. When instantiated in type-3 pairings $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q)$, let us assume that $c_1$ elements are in $\mathbb{G}_1$ and secret keys $\mathsf{sk}_{\mathsf{id}}$ are in $\mathbb{G}_2$. Note that, $\mathsf{FO}_{\mathsf{IBE}}.\mathsf{Enc}(\mathsf{mpk}, \mathsf{msg})$ would produce a ciphertext $c = (g_1^{H(r,\mathsf{msg})}, H'(g_{2,\mathsf{id}}^{H(r,\mathsf{msg})}) \oplus r, G(r) \oplus \mathsf{msg})$, where $H : \{0,1\}^* \times \{0,1\}^* \to \mathbb{Z}_q$, $H' : \mathbb{G}_T \to \{0,1\}^{\ell_1}$, $G : \{0,1\}^* \to \{0,1\}^{\ell_2}$. As for $\Pi_{\mathsf{FOAE}}$, we can use Elligator Squared [Tib14] as the pseudo-random encoding and considering BN curves [BN06] for instantiating the type-3 pairing we have $\mathbb{G}_1 = \mathbb{E}(\mathbb{F}_p)$ is a cyclic group of prime order $q$. Moreover, if we again set $\ell_1 = \ell_2 = \lambda$, then we end up with a 4-message anamorphic extension.

**Remark 3 (Anamorphism Notifier for IBAE)** We remark that while our formalisation of SCAN (in Section 7) is carried out in the context of AAE, the signaling technique is more general. In particular, one can design an analogous identification protocol for IBAE by applying the same PRF-based signaling strategy to generate the randomness used in encryption.

# 6 $\mu$-Message Asymmetric Anamorphic Signature Extension

In this section, we formalize the notion of $\mu$-message anamorphic extensions of signature schemes.

**Definition 26 ($\mu$-Message AAS Extension)** *A tuple* $\Pi_{\mathsf{AASE}} = (\mathsf{aGenS}, \mathsf{aGenR}, \mathsf{aSig}, \mathsf{aDec})$ *of PPT algorithms is a $\mu$-message asymmetric anamorphic signature extension ($\mu$-AASE) with anamorphic message*

*space* $\widehat{\mathcal{M}}$ *associated with a signature scheme* $\Pi_{\mathsf{SS}} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Verify})$ *with normal message space* $\mathcal{M}$ *and is defined as follows, where* $\mathsf{gp} \leftarrow \mathsf{SS.Setup}(1^\lambda)$*:*

$\underline{(\mathsf{ask}, \mathsf{avk}, \mathsf{td}) \leftarrow \mathsf{aGenS}(\mathsf{gp}) :}$ *The anamorphic key generation algorithm* aGenS *is executed by the sender of the signature, takes as input the global parameters* gp *generated by* Setup *of the associated signature scheme, and outputs a signature key pair* $(\mathsf{ask}, \mathsf{avk})$[10] *as well as some secret trapdoor information* td*.*

$\underline{(\mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGenR}(\mathsf{gp}) :}$ *The anamorphic key generation algorithm* aGenR *is executed by the receiver of the signature, takes as input the global parameters* gp *generated by* Setup *of the associated signature scheme, and outputs a double key pair* $(\mathsf{dk}, \mathsf{tk})$*.*

$\underline{\mathbf{asig} \leftarrow \mathsf{aSig}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg}) :}$ *The anamorphic signing algorithm* aSig *takes as input the sender's signing key* ask *with associated trapdoor information* td*, and the receiver's encryption double key* dk*, a regular message vector* $\mathbf{msg} \in \mathcal{M}^\mu$ *and an anamorphic message* $\mathsf{amsg} \in \widehat{\mathcal{M}}$*, and outputs an anamorphic signature vector* **asig***.*

$\underline{\mathsf{amsg} \leftarrow \mathsf{aDec}(\mathsf{tk}, \mathsf{avk}, \mathbf{asig}, \mathbf{msg}) :}$ *The anamorphic decryption algorithm takes as input the receiver's secret decryption double key* tk*, the sender's signature verification key* avk*, and an anamorphic signature vector* **asig** *along with a message vector* **msg***, and outputs an anamorphic message* amsg *or* $\perp$*.*

*It must satisfy the following properties.*

**Correctness**: *For all* $\lambda \in \mathbb{N}$, $\mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathbf{msg}, \mathsf{amsg}) \in \mathcal{M}^\mu \times \widehat{\mathcal{M}}$*, it holds*

$$\Pr\left[\mathsf{amsg} \neq \mathsf{aDec}(\mathsf{tk}, \mathsf{avk}, \mathbf{asig}, \mathbf{msg}) \;:\; \begin{array}{c}(\mathsf{ask}, \mathsf{avk}, \mathsf{td}) \leftarrow \mathsf{aGenS}(\mathsf{gp}) \\ (\mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGenR}(\mathsf{gp}) \\ \mathbf{asig} \leftarrow \mathsf{aSig}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg})\end{array}\right] \leq \mathsf{negl}(\lambda)$$

*where the probability is taken over the random coins of* aGenS, aGenR *and* aSig*.*

**Anamorphic security**: *For every PPT dictator* $\mathcal{D}$ *it holds that*

$$\left|\Pr[\mathsf{GReal}^{\mathcal{D}}_{\mathsf{SS}}(\lambda) = 1] - \Pr[\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{AASE}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security games are defined as follows:*

| $\underline{\mathsf{GReal}^{\mathcal{D}}_{\mathsf{SS}}(\lambda) :}$ | $\underline{\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{AASE}}(\lambda) :}$ |
|---|---|
| 1. $\mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda)$ | 1. $\mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda)$ |
| 2. $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(\mathsf{gp})$ | 2. $(\mathsf{ask}, \mathsf{avk}, \mathsf{td}) \leftarrow \mathsf{aGenS}(\mathsf{gp})$ |
| 3. ***return*** $\mathcal{D}^{\mathsf{OSig}(\cdot,\cdot)}(\mathsf{sk}, \mathsf{vk})$ | 3. $(\mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGenR}(\mathsf{gp})$ |
| *where* $\mathsf{OSig}(\mathbf{msg}, \mathsf{amsg}) :=$ | 4. ***return*** $\mathcal{D}^{\mathsf{OaSig}(\cdot,\cdot)}(\mathsf{ask}, \mathsf{avk})$ |
| $\mathsf{Sig}(\mathsf{sk}, \mathbf{msg}) := (\mathsf{Sig}(\mathsf{sk}, \mathsf{msg}_i))_{i \in [\mu]}$. | *where* $\mathsf{OaSig}(\mathbf{msg}, \mathsf{amsg}) :=$ |
| | $\mathsf{aSig}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg})$. |

**Preserving EUF/SUF-CMA security.** We note that anamorphic security clearly also holds for an adversary not having access to ask. Such an adversary does not notice whether it interacts with the regular signature scheme or the anamorphic version. This implies that the EUF-CMA or SUF-CMA security of $\Pi_{\mathsf{SS}}$

[10] We assume that gp is included in vk and avk. It is also available to all the algorithms.

carries over to the individual signatures of the anamorphic extension.

**Choice of gp.** As opposed to AAE, the benefits of deploying *asymmetric* cryptography to realize the anamorphic channel for signatures would be highly reduced if each and every signer chose its own unique $\mathsf{gp}$. In this case, a receiver of an anamorphic message needs to generate an individual key pair $(\mathsf{dk}, \mathsf{tk})$ for each signer it wants to receive anamorphic messages from. To this end, we make the realistic assumption that signers only choose from a small set of $\mathsf{gp}$ settings, e.g., standardized elliptic curves, or the dictator fixes some $\mathsf{gp}$ for everyone.

**IND-CPA security of anamorphic signatures.** Similar to AAE, another natural security property for an anamorphic signature scheme is to guarantee that within a group of people that communicate via this embedded covert channel, no information about a message is leaked to parties other than the intended receiver. This is captured by the following IND-CPA-style security definition.

**Definition 27 (IND-CPA Security for AASE)** *A $\mu$-AASE scheme $\Pi_{\mathsf{AASE}}$ associated with signature scheme $\Pi_{\mathsf{SS}}$ is said to satisfy IND-CPA security if for every PPT adversary $\mathcal{A}$ it holds that*

$$\left|\Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AASE}}(\lambda, 1) = 1] - \Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AASE}}(\lambda, 0) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security game is defined as follows:*

$\underline{\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{AASE}}(\lambda, \beta)}$ :

1. $\mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{ask}, \mathsf{avk}, \mathsf{td}) \leftarrow \mathsf{aGenS}(\mathsf{gp})$
3. $(\mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGenR}(\mathsf{gp})$
4. $(\mathbf{msg}, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}^{\mathsf{OaSig}(\cdot,\cdot)}(\mathsf{avk}, \mathsf{dk})$
5. $\mathbf{asig}_\beta \leftarrow \mathsf{aSig}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg}_\beta)$
6. ***return*** $\beta' \leftarrow \mathcal{A}^{\mathsf{OaSig}(\cdot,\cdot)}(\mathsf{asig}_\beta)$ *where*

   $\mathsf{OaSig}(\mathbf{msg}, \mathsf{amsg}) := \mathsf{aSig}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg})$.

**(s)IND-(R)CCA security of anamorphic signatures.** Similar to AAE, we can also define stronger security properties for anamorphic signatures by providing a (restricted) $\mathsf{aDec}$ oracle or $\mathsf{ask}$ as input to $\mathcal{A}$. So we define IND-CCA security by providing an $\mathsf{aDec}(\mathsf{tk}, \mathsf{avk}, \cdot, \cdot)$ oracle (rejecting if queried with challenge $(\mathbf{asig}_\beta, \mathbf{msg})$), sIND-CCA by additionally giving $\mathsf{ask}$ as input to $\mathcal{A}$, and sIND-RCCA like sIND-CCA but restricting the oracle to reject if $\mathsf{aDec}(\mathsf{tk}, \mathsf{avk}, \mathbf{asig}, \mathbf{msg}) \in \{\mathsf{amsg}_0, \mathsf{amsg}_1\}$.

## 6.1 Anamorphism-Friendly Signatures with Compliant Encryption

Generalizing the ideas from Waters signature with linear encryption (Section 6.2) and BBS with ElGamal and pseudorandom encodings (Section 6.3), the following definition covers signature schemes which are canonical candidates for an anamorphic extension. The signatures generated by these schemes have at least one uniformly (and freshly) chosen component and allow the rest of the signature to be computed (cf. $\mathsf{RemSig}$ below) by treating the uniform components in a black-box way, that means, without using additional information related to the generation of these components (e.g., the DL $d$ used to create a random group element $g^d$). To enable the latter, some trapdoor information from the key generation might be used (cf. $\mathsf{SS.Gen}'$ below). An encryption scheme is compliant to such a signature scheme, in the sense that it easily allows for the construction of an anamorphic extension, if its ciphertext components are pseudo-random (IND$-CPA, see Definition 10) and can be completely embedded in one or multiple signatures.

$\underline{\mathsf{aGenS}(\mathsf{gp})}$ :

1. $(\mathsf{ask}, \mathsf{avk}, \mathsf{td}) \leftarrow \mathsf{SS.Gen}'(\mathsf{gp})$
2. **return** $(\mathsf{ask}, \mathsf{avk}, \mathsf{td})$

$\underline{\mathsf{aGenR}(\mathsf{gp})}$ :

1. $(\mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{ES.Gen}(\mathsf{gp})$
2. **return** $(\mathsf{dk}, \mathsf{tk})$

$\underline{\mathsf{aDec}(\mathsf{tk}, \mathsf{avk}, \mathbf{asig}, \mathbf{msg})}$ :

0. **for** $0 \le i \le \mu - 1$: **if** $\mathsf{SS.Ver}(\mathsf{avk}, \mathsf{sig}_i, \mathsf{msg}_i) = \bot$ **return** $\bot$
1. **for** $0 \le i \le m-1$, $0 \le j \le e'_i - 1$:
   $c_{i,j} \coloneqq s_{i,j \bmod e_i}^{(\lfloor \frac{j}{e_i} \rfloor)}$
2. **return** $\mathsf{amsg} \leftarrow \mathsf{ES.Dec}(\mathsf{tk}, (c_i)_i)$

$\underline{\mathsf{aSig}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg})}$ :

1. Parse $\mathbf{msg} = (\mathsf{msg}_0, \ldots, \mathsf{msg}_{\mu-1})$
2. $\mathsf{act} = (c_0, \ldots, c_{m-1}) \leftarrow \mathsf{ES.Enc}(\mathsf{dk}, \mathsf{amsg})$
   where $c_i = (c_{i,0}, \ldots, c_{i,e_i-1}) \in S_i^{e'_i}$
3. **for** $0 \le i \le \mu - 1$, $0 \le \ell \le m-1$, $0 \le j \le e_\ell - 1$:
   $s_{\ell,j}^{(i)} \coloneqq \begin{cases} c_{\ell, ie_\ell + j}, & ie_\ell + j < e'_\ell \\ a \leftarrow S_\ell, & \text{else} \end{cases}$
4. **for** $0 \le i \le \mu - 1$:
   $r^{(i)} \leftarrow \mathsf{SS.RemSig}(\mathsf{ask}, \mathsf{td}, \mathsf{msg}_i, (s_j^{(i)})_j)$
   $\mathsf{sig}_i \coloneqq ((s_j^{(i)})_j, r^{(i)})$
5. **return asig** $\coloneqq (\mathsf{sig}_0, \ldots, \mathsf{sig}_{\mu-1})$

Figure 7: Generic $\mu$-AASE $\Pi_{\mathsf{aGAE}}$ and variant $\Pi_{\mathsf{aGAE}'}$

**Definition 28 (Anamorphism-Friendly Sign. & Compliant Enc.)** *Let $\Pi_{\mathsf{SS}}$ be a signature scheme with a signature space that can be written as $S_0^{e_0} \times \ldots \times S_{m-1}^{e_{m-1}} \times R$, where $S_i \neq S_j$ and $e_i > 0$, such that the following conditions hold:*

1. *Random components: $m - 1 \ge 0$ and for signatures $(s_0, \ldots, s_{m-1}, r)$ generated by $\mathsf{SS.Sig}$, it holds that all $s_i \leftarrow S_i^{e_i}$ are uniformly and freshly chosen.*
2. *Black-box computation of remainder: There exist two PPT algorithms $\mathsf{SS.Gen}'$, $\mathsf{SS.RemSig}$ such that $(\mathsf{sk}, \mathsf{vk}) \approx_c (\mathsf{sk}', \mathsf{vk}')$, where $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SS.Gen}(\mathsf{gp})$ and $(\mathsf{sk}', \mathsf{vk}', \mathsf{td}) \leftarrow \mathsf{SS.Gen}'(\mathsf{gp})$, and*

$$\Pr\left[ b = b' : \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{SS.Setup}(1^\lambda) \\ (\mathsf{sk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{SS.Gen}'(\mathsf{gp}) \\ \mathsf{msg} \leftarrow \mathcal{A}(\mathsf{sk}, \mathsf{vk}) \\ (s_0, \ldots, s_{m-1}, r_0) \leftarrow \mathsf{SS.Sig}(\mathsf{sk}, \mathsf{msg}) \\ r_1 \leftarrow \mathsf{SS.RemSig}(\mathsf{sk}, \mathsf{td}, \mathsf{msg}, (s_0, \ldots, s_{m-1})) \\ b \leftarrow \{0,1\} \\ b' \leftarrow \mathcal{A}(s_0, \ldots, s_{m-1}, r_b) \end{array} \right] \le \frac{1}{2} + \mathsf{negl}(\lambda)$$

*In this case, we call $\Pi_{\mathsf{SS}}$ anamorphism-friendly. We call an encryption scheme $\Pi_{\mathsf{ES}}$ compliant to $\Pi_{\mathsf{SS}}$ if it is IND$-CPA secure and its ciphertext space can be written as $S_0^{e'_0} \times \ldots \times S_{m-1}^{e'_{m-1}}$ with $e'_i > 0$.*

In this situation, we can securely embed a ciphertext containing the anamorphic message into the components of one or multiple signatures. This is specified by the aSig algorithm in Figure 7. In Step 3, as many ciphertext components as possible are packed into the same signature. As it may happen that the designated signature components cannot be completely "filled" with ciphertext components, we use truly random elements in this case. The remainder of the signature is computed in Step 4 using RemSig. When receiving $\mu$ signatures, the ciphertext components are first unpacked in Step 2 of aDec before amsg is recovered. By following this blueprint using an anamorphism-friendly signature and compliant encryption, we

obtain a secure anamorphic extension $\Pi_{\mathsf{aGAE}}$ also satisfying IND-CPA security. If the underlying encryption additional satisfies IND-RCCA security then $\Pi_{\mathsf{aGAE}}$ achieves sIND-RCCA security. Note that an IND-CCA secure PKE still leads to an sIND-RCCA but not IND-CCA secure extension when only following Steps 1 and 2 of of $\mathsf{aDec}$ in Figure 7 as the adversary (against the extension) may tamper with the $r^{(i)}$ of the challenge which would lead to a different **asig** decrypting to $\mathsf{amsg}_\beta$. To additionally achieve IND-CCA security, $\mathsf{aDec}$ needs to verify the signatures **asig** (Step 0 in Figure 7) and the signature scheme needs to be SUF-CMA secure. Still sIND-CCA security cannot be achieved in this way as, given $\mathsf{ask}$, the adversary may reuse the challenge $c_{i,j}$ to generate a new signature. We now formally prove the security in the following theorem.

**Theorem 7** *Let $\Pi_{\mathsf{SS}}$ be an anamorphism-friendly signature scheme and $\Pi_{\mathsf{ES}}$ an encryption scheme compliant to $\Pi_{\mathsf{SS}}$. Then $\Pi_{\mathsf{aGAE}}$ as defined in Figure 7 is a $\mu$-message anamorphic signature extension, where $\mu = \max\{\lceil \frac{e'_i}{e_i} \rceil\}$, satisfying anamorphic and IND-CPA security. If $\Pi_{\mathsf{ES}}$ additionally achieves IND-RCCA security then $\Pi_{\mathsf{aGAE}}$ achieves sIND-RCCA security. Moreover, if $\Pi_{\mathsf{ES}}$ satisfies IND-CCA security and $\Pi_{\mathsf{SS}}$ satisfies SUF-CMA security, then the variant $\Pi_{\mathsf{aGAE}'}$ in Figure 7 achieves IND-CCA security.*

**Proof.** We prove this theorem in two parts.

**Anamorphic Security.** In the following sequence of hybrid games let $\mathcal{D}$ be a PPT dictator and $Q$ its total number of oracle queries.

Game 0**:** This is the real game $\mathsf{GReal}^{\mathcal{D}}_{\mathsf{SS}}(\lambda)$ in the anamorphic security definition, where the challenger generates $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SS.Gen}(\mathsf{gp})$ and handles signature queries $\mathsf{OSig}(\mathbf{msg}, \mathsf{amsg})$ of the dictator by computing $(\mathsf{SS.Sig}(\mathsf{sk}, \mathsf{msg}_i)$ for $0 \leq i \leq \mu - 1$.

Game 1**:** In this game the challenger generates the key pair by computing $(\mathsf{ask}, \mathsf{avk}, \mathsf{td}) \leftarrow \mathsf{SS.Gen}'(\mathsf{gp})$ and provides $(\mathsf{ask}, \mathsf{avk})$ as input to $\mathcal{D}$.

Game $(2, j)$ **for** $j \in [0, \mu Q]$**:** In this series of games we slightly modify Game 1. We define Game $(2, 0)$ to be identical to Game 1. In Game $(2, j)$, the challenger handles the first $j$ signature computations (in order to respond to oracle calls) as follows: It computes $(s_0, \ldots, s_{m-1}, r) \leftarrow \mathsf{SS.Sig}(\mathsf{ask}, \mathsf{msg}_j)$ but replaces $r$ with $r' \leftarrow \mathsf{SS.RemSig}(\mathsf{ask}, \mathsf{td}, \mathsf{msg}_j, (s_0, \ldots, s_{m-1}))$, i.e., it returns $(s_0, \ldots, s_{m-1}, r')$. The remaining signature computations are left untouched in comparison to Game $(2, 0)$. Note that in Game $(2, \mu Q)$ we have modified all signature computations in this way.

Game 3**:** In this game, we modify all signature computations done by the challenger (in comparison to Game $(2, \mu Q)$) at once: instead of calling $\mathsf{SS.Sig}$ to compute $(s_0, \ldots, s_{m-1})$ being input to $\mathsf{SS.RemSig}$, we choose them uniformly at random, i.e., $(s_0, \ldots, s_{m-1}) \leftarrow S_0^{e_0} \times S_{m-1}^{e-1}$.

Game $(4, j)$ **for** $j \in [0, Q]$**:** We define Game $(4, 0)$ to be identical to Game 3, except that the challenger additionally generates the key pair $(\mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{ES.Gen}(\mathsf{gp})$ of the encryption scheme. In Game $(4, j)$, the challenger handles the first $j$ oracle calls $\mathsf{OSig}(\mathbf{msg}, \mathsf{amsg})$ as follows: it computes $(c_0, \ldots, c_{m-1}) \leftarrow \mathsf{ES.Enc}(\mathsf{dk}, \mathsf{amsg})$ and embeds the ciphertext components into $(s_0, \ldots, s_{m-1})$ as defined by Steps 2 and 3 of the $\mathsf{aSig}$ definition in Figure 7. Then $\mathsf{SS.RemSig}$ is called as in Game 3. Note that in Game $(4, Q)$ we have embedded ciphertexts in all responses to oracle calls.

Game 5**:** This is the anamorphic game $\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{GAE}}(\lambda)$, where the challenger generates the keys as $(\mathsf{ask}, \mathsf{avk}, \mathsf{td}) \leftarrow \mathsf{GAE.aGenS}(\mathsf{gp})$, $(\mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{GAE.aGenR}(\mathsf{gp})$ and handles signature queries $\mathsf{OaSig}(\mathbf{msg}, \mathsf{amsg})$ of the dictator by computing $\mathsf{aSig}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg})$.

The indistinguishability of Game 0 and Game 1 follows immediately from the second property of an anamorphism-friendly signature. Game 1 and Game $(2, 0)$ are identical.

Game $(2, 0)$ is indistinguishable from Game $(2, Q)$, as a distinguisher $\mathcal{D}$ for Game $(2, j)$ and Game $(2, j+1)$ (for random $j \in \{0, \ldots, \mu Q - 1\}$) can be turned in to an adversary against the black-box computation of remainder property of an anamorphism-friendly signature. Game $(2, \mu Q)$ is perfectly indistinguishable from Game 3 due to the random component property of the signature scheme. Game 3 and Game $(4, 0)$ are

perfectly indistinguishable as the modification is purely conceptional. Game $(4,0)$ is indistinguishable from Game $(4,Q)$, as from a distinguisher $\mathcal{D}$ for Game $(4,j)$ and Game $(4,j+1)$ (for random $j \in \{0,\ldots,Q-1\}$) an adversary against the IND$-CPA security of $\Pi_{\mathsf{ES}}$ can be constructed. Finally, Game $(4,Q)$ and Game 5 are identical.

**IND-CPA Security of $\Pi_{\mathsf{aGAE}}$.** This follows immediately from the IND$-CPA security of $\Pi_{\mathsf{ES}}$: Let Game 0 be the IND-CPA game for $\beta = 1$. In Game 1, when computing the challenge signature $\mathsf{asig}_1 \leftarrow \mathsf{aSig}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg}_1)$, we replace the ciphertexts $(c_0,\ldots,c_{m-1})$ to be embedded by aSig by randomly chosen values from the ciphertext space $S_0^{e_0} \times \ldots \times S_{m-1}^{e_{m-1}}$. Game 0 and Game 1 are indistinguishable if $\Pi_{\mathsf{ES}}$ is IND$-CPA. Note that the reduction algorithm can handle signature queries as it can choose $\mathsf{ask}, \mathsf{td}$ itself. In Game 2, we immediately revert this change by embedding ciphertexts encrypting $\mathsf{amsg}_0$, i.e., the challenger computes $\mathsf{asig}_0 \leftarrow \mathsf{aSig}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg}_0)$. Again, Game 1 and Game 2 are indistinguishable if $\Pi_{\mathsf{ES}}$ is IND$-CPA. Game 2 is the IND-CPA game for $\beta = 0$.

**sIND-RCCA Security of $\Pi_{\mathsf{aGAE}}$.** This follows immediately from the assumed IND-RCCA security of $\Pi_{\mathsf{ES}}$. The reduction algorithm $\mathcal{B}$ is given $\mathsf{gp}, \mathsf{dk}$ as input from its IND-RCCA challenger and has access to a $\mathsf{Dec}(\mathsf{tk},\cdot)$ oracle. It can generate $(\mathsf{avk}, \mathsf{ask}, \mathsf{td}) \leftarrow \mathsf{aGenR}(\mathsf{gp})$ itself, and thus provide $(\mathsf{avk}, \mathsf{dk}, \mathsf{ask})$ as input to the sIND-RCCA adversary $\mathcal{A}$. Clearly, $\mathcal{B}$ can handle $\mathcal{A}$'s $\mathsf{aSig}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \cdot, \cdot)$ oracle queries since it knows all necessary inputs. $\mathcal{B}$ can also handle $\mathsf{aDec}(\mathsf{tk}, \mathsf{avk}, \cdot, \cdot)$ queries by extracting the $c_{i,j}$ from the given **asig** as specified in Figure 7 and querying its own $\mathsf{Dec}(\mathsf{tk},\cdot)$ oracle this input. Clearly, this simulated aDec oracle behaves exactly as the real aDec oracle (in particular it rejects an input exactly when the real one does). To generate the challenge signature for the given $(\mathbf{msg}, \mathsf{amsg}_0, \mathsf{amsg}_1)$, $\mathcal{B}$ provides $(\mathsf{amsg}_0, \mathsf{amsg}_1)$ to its own challenger and embeds the response into signatures for **msg** as specified by Steps 3 to 5 of aSig in Figure 7. Finally $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

**IND-CCA Security of $\Pi_{\mathsf{aGAE}'}$.** This proof is similar to the sIND-RCCA security proof for $\Pi_{\mathsf{aGAE}}$ we have seen above, where the SUF-CMA security of $\Pi_{\mathsf{SS}}$ now ensures that the simulation of the aDec using a IND-CCA Dec oracle works out.

So let us consider how the reduction algorithm $\mathcal{B}$ (the adversary against the IND-CCA security of $\Pi_{\mathsf{ES}}$) handles $\mathcal{A}$'s queries of the $\mathsf{aDec}(\mathsf{tk}, \mathsf{avk}, \cdot, \cdot)$ IND-CCA oracle: For given $(\mathbf{asig}, \mathbf{msg})$, it first checks whether $(\mathbf{asig}, \mathbf{msg})$ equals the challenge signature and message vectors $(\mathbf{asig}^*, \mathbf{msg}^*)$ and rejects if it does. Then it verifies each of the $\mu$ signatures contained **asig** and returns $\perp$ if at least one of these checks fail. Otherwise, it extracts the $c_{i,j}$ from the given **asig** as specified in Figure 7 and queries its own $\mathsf{Dec}(\mathsf{tk},\cdot)$ oracle for this input. Obviously, the only thing this simulation does differently compared to a real aDec oracle is calling the $\mathsf{Dec}(\mathsf{tk},\cdot)$ oracle instead of executing the $\mathsf{Dec}(\mathsf{tk},\cdot)$ algorithm. This can lead to the following simulation failure: The $\mathsf{Dec}(\mathsf{tk},\cdot)$ oracle on input of the challenge ciphertext $c^* = (c^*_{i,j})_{i,j}$ (generated by $\mathcal{B}$'s IND-CCA challenger) will reject while the algorithm will return the corresponding decryption result. Hence, we need to show that the checks (particularly the signature verification) performed before this point ensure that the $\mathsf{Dec}(\mathsf{tk},\cdot)$ oracle will never be queried with $c^*$ as input.

Let us consider the different inputs $(\mathbf{asig}, \mathbf{msg})$ to the aDec oracle which could lead to $c^*$:

1. $(\mathbf{asig}, \mathbf{msg}) = (\mathbf{asig}^*, \mathbf{msg}^*)$: In this case $\mathcal{B}$ immediately rejects.
2. $\mathbf{asig} = \mathbf{asig}^*$ and $\mathbf{msg} \neq \mathbf{msg}^*$: In this case, we have at least one verifying signature $\mathsf{sig}_i^*$ for a non-challenge message $\mathsf{msg}_i \neq \mathsf{msg}_i^*$. There is only a negligible chance that a call to the aSig oracle (including $\mathsf{msg}_i$ as part of the input) could have produced the same signature since the used randomness for producing this signature is freshly chosen. In particular, if we write $\mathsf{sig}_i^*$ as $\mathsf{sig}_i^* = ((s_j^{(i)}, r^{(i)})$ as in Figure 7, for the newly produced signature the same $s_j^{(i)}$ would need to be chosen. Hence, in this case the SUF-CMA security is broken and thus we could build a adversary against the underlying signature scheme.

3. $\mathbf{asig} \neq \mathbf{asig}^*$ and $\mathbf{msg} = \mathbf{msg}^*$: In this case, we have at least one signature $\mathsf{sig}_i \neq \mathsf{sig}_i^*$ which verifies for the challenge message $\mathsf{msg}_i^*$. Since also with this signature $\mathsf{sig}_i$, the extraction of the ciphertext need to yield $c^*$ which is composed of the $s_j^{(i)}$, it holds that at least one of the random components of $\mathsf{sig}_i$ needs to coincide with the one of $\mathsf{sig}_i^*$. Similar to the previous case, we can argue that such $\mathsf{sig}_i$ cannot stem from a call to the $\mathsf{aSig}$ oracle and thus $\mathsf{sig}_i$ is a forgery of a new signature for the challenge message $\mathsf{msg}_i^*$. Thus, in this case the SUF-CMA security is broken.

4. $\mathbf{asig} \neq \mathbf{asig}^*$ and $\mathbf{msg} \neq \mathbf{msg}^*$: In this case, we are either in one of the two previous situations (for which we have already shown that they violate SUF-CMA security) or for all $i$ it holds that $\mathsf{sig}_i \neq \mathsf{sig}_i^*$ and $\mathsf{msg}_i \neq \mathsf{msg}_i^*$. So let us consider the latter. Using the same argument as in the previous case, it follows that there is a negligible probability that $\mathsf{sig}_i$ stems from a call to the $\mathsf{aSig}$ oracle and thus we either have a signature forgery for a new message or a message for which the $\mathsf{aSig}$ oracle has been queried before. Both cases violate the SUF-CMA security of the signature scheme.

To summarize, assuming SUF-CMA security, the IND-CCA adversary $\mathcal{B}$ against $\Pi_{\mathsf{ES}}$ can perfectly simulate the $\mathsf{aDec}$ oracle for $\mathcal{A}$ and thus $\mathcal{A}$'s IND-CCA challenger for $\Pi_{\mathsf{aGAE}}$. Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

To make the above argument more formal, one can define a sequence of games, where the first game is the IND-CCA game for $\Pi_{\mathsf{aGAE}'}$ for $\beta = 1$ and the last game is the IND-CCA game for $\Pi_{\mathsf{aGAE}'}$ for $\beta = 0$. In the first three intermediate games, we modify the $\mathsf{aDec}$ oracle by considering the cases 2-4 from above and let the oracle reject if the corresponding event happens. Indistinguishability between the games can then be shown using the SUF-CMA security of $\Pi_{\mathsf{SS}}$ (based on the arguments made above). After that, we can be sure that an $\mathsf{aDec}$ oracle can be simulated by a $\mathsf{Dec}$ oracle and thus we can change the challenge message to be encrypted from $\mathsf{amsg}_1$ to $\mathsf{amsg}_0$ in the next and final game. Here indistinguishability follows from the IND-CCA security for $\Pi_{\mathsf{ES}}$. □

Next, we instantiate our generic AASE using (a) Waters signatures with linear encryption presented in Section 6.2, and (b) BBS with the encryption scheme obtained by combining ElGamal with a suitable pseudorandom encoding in Section 6.3. The security follows from the master Theorem 7.

## 6.2 Waters Signatures with Linear Encryption

We present an AASE construction based on Water's signatures [Wat05] and Linear Encryption (LE) [BBS04]. LE is a variant of ElGamal relying on the Decision linear ($\mathsf{DLIN}$) assumption that can be instantiated in a group $\mathbb{G}$ where the $\mathsf{DDH}$ assumption is easy, e.g., in symmetric pairing groups. We consider the Waters scheme over symmetric pairings groups $(\mathbb{G}, \mathbb{G}_T, g, e, q)$ for simplicity, but it can straightforwardly be translated to an asymmetric pairing setting (e.g., type-3 pairings). However, we note that the anamorphic extension still requires LE.[11]

We first recall that Waters signatures are pairing-based signatures $\mathsf{EUF\text{-}CMA}$ secure under the $\mathsf{CDH}$ assumption. They rely on the Waters hash $H(\mathsf{msg}) := h_0 \prod_{i=1}^{n} h_i^{m_i}$, where $m_i$ denotes the $i$'th bit of message $\mathsf{msg} \in \{0,1\}^n$ and $h_i \leftarrow \mathbb{G}$. A signature consists of $(\sigma_1, \sigma_2) := (g^r, g^{\alpha\beta} H(\mathsf{msg})^r)$ for random $r \in \mathbb{Z}_q$, secret key $g^{\alpha\beta}$ and public key $(g^\alpha, g^\beta, h_0, \ldots, h_n)$. A signature is verified by checking the equality $e(g^\alpha, g^\beta) \cdot e(\sigma_1, H(\mathsf{msg})) = e(g, \sigma_2)$. Also recall that in LE, the public key is $(u, v, h) \in \mathbb{G}^3$ and the secret key $(x, y) \in \mathbb{Z}_q$ such that $u^x = v^y = h$ holds. Encrypting a message $\mathsf{msg} \in \mathbb{G}$ amounts to choosing random $\kappa_1, \kappa_2$ and computing ciphertext $(c_1, c_2, c_2) := (u^{\kappa_1}, v^{\kappa_2}, h^{\kappa_1+\kappa_2} \cdot \mathsf{msg})$ and decryption computes $\mathsf{msg} = c_3 \cdot (c_1^x \cdot c_2^y)^{-1}$. LE is IND-CPA secure under DLIN.

The essential idea is to embed a ciphertext $(c_1, c_2, c_3)$ of an anamorphic message $\mathsf{amsg}$ as the first components of three Waters signatures as this component is uniformly chosen from $\mathbb{G}$ and a LE ciphertext is indistinguishable from a random $\mathbb{G}^3$ element under $\mathsf{DLIN}$ (cf. IND$-CPA, Definition 10). However, we face the problem that the discrete logarithm $r$ of this component is needed to compute the second component of the signature and for the $c_i$ the DLs are unknown. To circumvent this problem, we make the assumption

[11] We cannot rely on DDH as elements of the ElGamal ciphertext would be required in both source groups.

$\underline{\mathsf{aGenS}(\mathsf{gp})}$ :

1. Parse $\mathsf{gp} = (\mathbb{G}, \mathbb{G}_T, e, g, g_T, p)$
2. $\alpha, \beta, a_0, \ldots, a_n \leftarrow \mathbb{Z}_p$
3. $h_0 := g^{a_0}, \ldots, h_n := g^{a_n}$ and $H : \{0,1\}^n \to \mathbb{G}$ with $\mathsf{msg} \mapsto h_0 \prod_{i=1}^{n} h_i^{m_i}$
4. $\mathsf{ask} := g^{\alpha\beta}$, $\mathsf{avk} := (g^\alpha, g^\beta, h_0, \ldots, h_n)$, $\mathsf{td} := (a_0, \ldots, a_n)$
5. **return** $(\mathsf{ask}, \mathsf{avk}, \mathsf{td})$

$\underline{\mathsf{aGenR}(\mathsf{gp})}$ :

1. Parse $\mathsf{gp} = (\mathbb{G}, \mathbb{G}_T, e, g, g_T, p)$
2. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{LinE.Gen}(\mathsf{gp})$

   // $a, b, x \leftarrow \mathbb{Z}_p$, and $y := axb^{-1}$

   // $u := g^a$, $v := g^b$, $h := g^{ax}$

   // $\mathsf{pk} := (u, v, h)$, $\mathsf{tk} := (x, y)$
3. $\mathsf{dk} := \mathsf{pk}$, $\mathsf{tk} := \mathsf{sk}$
4. **return** $(\mathsf{dk}, \mathsf{tk})$

$\underline{\mathsf{aSig}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg})}$ :

1. Parse $\mathbf{msg} = (\mathsf{msg}_1, \mathsf{msg}_2, \mathsf{msg}_3)$
2. $(c_1, c_2, c_3) \leftarrow \mathsf{LinE.Enc}(\mathsf{dk}, \mathsf{amsg})$

   // $\kappa_1, \kappa_2 \leftarrow \mathbb{Z}_p$

   // $c_1 = u^{\kappa_1}$, $c_2 = v^{\kappa_2}$,

   // $c_3 = h^{\kappa_1+\kappa_2} \cdot \mathsf{amsg}$)
3. **for** $j \in \{1, 2, 3\}$:
   $\mathsf{sig}_j := (c_j, \mathsf{ask}\, c_j^{a_0} \prod_{i=1}^{\ell} c_j^{a_i \cdot m_{j,i}})$
4. **return** $\mathbf{asig} := (\mathsf{sig}_1, \mathsf{sig}_2, \mathsf{sig}_3)$

$\underline{\mathsf{aDec}(\mathsf{tk}, \mathsf{avk}, \mathbf{asig}, \mathbf{msg})}$ :

1. Parse $\mathbf{asig} = ((c_1, s_1), (c_2, s_2), (c_3, s_3))$
2. $\mathsf{amsg} \leftarrow \mathsf{LinE.Dec}(\mathsf{tk}, (c_1, c_2, c_3))$

   // $\mathsf{amsg} = c_3 \cdot (c_1^x \cdot c_2^y)^{-1}$
3. **return** $\mathsf{amsg}$

Figure 8: 3-AASE $\Pi_{\mathsf{aWaters}}$

that every user generates its own parameters for the Waters hash and thus knows $a_i$ such that $h_i = g^{a_i}$ for $i \in \{0, \ldots, n\}$ and that this is accepted by the dictator.[12] Using the $a_i$ allows us to compute the second signature component without knowing the DL $r$.

The resulting scheme is presented in Figure 8. We note that the same strategy can also be applied to the SUF-CMA secure variant by Boneh, Shen and Waters [BSW06]. The security of anamorphic signature extensions following this strategy is implied by our master theorem (Theorem 7) presented in Section 6.1, which is discussed below.

**Theorem 8** *Waters signatures (Figure 8) are anamorphism-friendly and linear encryption is compliant to them under the* DLIN *assumption.*

**Proof.** We prove the theorem in two parts.

**Anamorphism-Friendliness.** The signature space of Waters is $\mathbb{G}^2$ where $(\mathbb{G}, \mathbb{G}_T, e, p)$ is a bilinear group setting. In the notation of Defintion 28, we can write it as $S_0 \times R$, with $S_0 = \mathbb{G}$ and $R = \mathbb{G}$. Considering only the first component of a signature $(g^d, g^{\alpha\beta} H(\mathsf{msg})^d)$, we observe that it is uniformly and freshly as $d \leftarrow \mathbb{Z}_p$. Hence, the first property is satisfied. Regarding the second property, we observe that $g^{\alpha\beta} H(\mathsf{msg})^d = g^{\alpha\beta}(h_0 \prod_{i=1}^{\ell} h_i^{m_i})^d$, where $h_i = g^{a_i}$ are elements chosen during signature key generation, can alternatively be computed by setting $\mathsf{td} := (a_0, \ldots, a_n)$ and $\mathsf{RemSig}(\mathsf{ask}, \mathsf{td}, \mathsf{msg}, s = g^d) := \mathsf{ask}\, s^{a_0} \prod_{i=1}^{\ell} s^{a_i m_i}$. $\mathsf{Gen}'$ outputs a key pair with the same distribution as $\mathsf{Gen}$ and additionally $\mathsf{td}$. This implies the second property.

[12] We are aware that this is a somewhat strong assumption as this could be suspicious for the dictator. Our second AASE from BBS does not suffer from this limitation

$\underline{\mathsf{aGenS}(\mathsf{gp})}$ :

1. Parse $\mathsf{gp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, q, H)$
2. $x \leftarrow \mathbb{Z}_q$
3. $\mathsf{ask} := x$, $\mathsf{avk} := g_2^x$
4. **return** $(\mathsf{ask}, \mathsf{avk}, \mathsf{td})$

$\underline{\mathsf{aGenR}(\mathsf{gp})}$ :

1. Parse $\mathsf{gp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, q, H)$
2. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{El.Gen}(\mathsf{gp})$

   // $x \leftarrow \mathbb{Z}_q$

   // $\mathsf{pk} = g_1^x$, $\mathsf{tk} = x$
3. $\mathsf{dk} := \mathsf{pk}$, $\mathsf{tk} := \mathsf{sk}$
4. **return** $(\mathsf{dk}, \mathsf{tk})$

$\underline{\mathsf{aSig}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathbf{msg}, \mathsf{amsg})}$ :

1. Parse $\mathbf{msg} = (\mathsf{msg}_1, \mathsf{msg}_2, \mathsf{msg}_3, \mathsf{msg}_4)$
2. $c := \mathsf{El.Enc}(\mathsf{dk}, \mathsf{amsg})$

   // $\kappa \leftarrow \mathbb{Z}_q$

   // $c = (c_1, c_2) = (g_1^\kappa, \mathsf{dk}^\kappa \mathsf{amsg})$
3. $(s_1, s_2, s_3, s_4) := \mathsf{Encode}(c)$

   // In the negligible event that

   // $\mathsf{Encode}(c) = \perp$, we return to 2.
4. **for** $j \in \{1, 2, 3, 4\}$:
   $\mathsf{sig}_j := (s_j, (\mathsf{H}(\mathsf{msg}_j)^{\frac{1}{\mathsf{ask}+s_j}}))$
5. **return asig** $:= (\mathsf{sig}_1, \mathsf{sig}_2, \mathsf{sig}_3, \mathsf{sig}_4)$

$\underline{\mathsf{aDec}(\mathsf{tk}, \mathsf{avk}, \mathbf{asig}, \mathbf{msg})}$ :

1. Parse $\mathbf{asig} = ((s_1, s_1'), (s_2, s_2'), (s_3, s_3'), (s_4, s_4'))$
2. $c := \mathsf{Decode}((s_1, s_2), (s_3, s_4))$
3. if $c = \perp$ return $\perp$
4. $\mathsf{amsg} := \mathsf{EL.Dec}(\mathsf{tk}, c)$
5. **return** $\mathsf{amsg}$

Figure 9: 4-AASE $\Pi_{\mathsf{aBBS}}$

**Compliance of Linear Encryption.** It is easy to see that the linear encryption scheme as defined in Section 6.2 is IND$-CPA over the group $\mathbb{G}$ under the DLIN assumption. The DLIN problem asks an adversary to distinguish $h^{x+y}$ from $h^z$, where $z \leftarrow \mathbb{Z}_p$, given $(u, v, h, u^x, v^y)$, where $u, v, h \leftarrow \mathbb{G}$ and $x, y \leftarrow \mathbb{Z}_p$. First we observe that the public key $(u, v, h)$ of the linear encryption scheme can be generated by choosing $a, b, x \leftarrow \mathbb{Z}_p$ and setting $y := axb^{-1}$, $u := g^a$, $v := g^b$, $h := g^{ax}$. In this way, $(u, v, h)$ are indeed random generators as in the DLIN problem and they satisfy $u^x = v^y = h$ as needed for decryption. Now, the reduction to the DLIN assumption is very simple: given an instance of the DLIN problem $(u, v, h, u^x, v^y, w)$, we compute the challenge ciphertext for the IND$-CPA adversary as $(c_1, c_2, c_3) = (u^x, v^y, w \cdot \mathsf{msg})$ and return the output of the adversary as our guess. Clearly, if $w$ is random then also the ciphertext, if $w = h^{x+y}$ then it is a proper encryption of msg. Finally, observe that the ciphertext space of the linear encryption is $\mathbb{G}^3$ and is thus compliant with Waters signature scheme. □

## 6.3 BBS with ElGamal and Pseudorandom Encodings

BBS signatures, as introduced in [BBS04], are a very well-known signature scheme and currently in the process of *standardization* by the IETF [LKWL24]. A formal proof of its SUF-CMA security under the $k$-SDH assumption in the standard model has only recently been given in [TZ23]. We consider BBS defined in asymmetric (type-3) pairing groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q)$ of prime order $q$. The public key is $g_2^x$ for secret

key $x \in \mathbb{Z}_q$. For a signature on message $\mathsf{msg} \in \mathbb{Z}_q^\ell$, one samples a random $d \in \mathbb{Z}_q$ and computes $\sigma = (\sigma_1, \sigma_2) = (d, H(\mathsf{msg})^{\frac{1}{x+d}}) \in \mathbb{Z}_q \times \mathbb{G}_1$. Verification is straightforward by checking whether $e(\sigma_2, g_2) = e(H(\mathsf{msg}), g_2^{\sigma_1} \cdot g_2^x)$. Regarding our construction, we may omit the detail that the hash $H(\mathsf{msg}) := g_1 \cdot \prod_{i=1}^{\ell} h_i^{\mathsf{msg}_i}$ is as in Waters scheme (though with $\mathsf{msg}_i \in \mathbb{Z}_q$) as this irrelevant. We set $\ell = 1$ in the following.

The basic idea (to obtain an IND-CPA secure extension) is to apply ElGamal encryption over $\mathbb{G}_1$ and to embed a ciphertext, which is pseudorandom over $\mathbb{G}_1^2$ under the DDH assumption, into a random component of two BBS signatures. The first component $d$ of such a signature is randomly chosen, but unfortunately over $\mathbb{Z}_q$ instead of $\mathbb{G}_1$. To this end, we need to specify a pseudorandom $(\mathbb{G}_1^2, \mathbb{Z}_q^\mu)$-encoding. Again, we can deploy Elligator Squared [Tib14] which also provides a suitable encoding for pairing-friendly curves $\mathbb{E}(\mathbb{F}_p)$ to $\mathbb{F}_p^2$. For instance, let us consider BN curves [BN06] for instantiating the asymmetric pairing-based setting of our BBS signatures. Then $\mathbb{G}_1 = \mathbb{E}(\mathbb{F}_p)$ is a cyclic group of prime order $q$. Note that for a random $h \in \mathbb{G}_1$, Elligator Squared will return a close to random element from $\mathbb{F}_p^2$ but not necessarily $\mathbb{F}_q^2$ as desired. However, since $p$ is very close to $q$, more precisely $p - q \approx \sqrt{p}$ (cf. Hasse's bound), the probability, assuming the standard element representation for $\mathbb{F}_p$ and $\mathbb{F}_q$ as $\{0, \ldots, p-1\}$ and $\{0, \ldots, q-1\}$, that we hit an element which is not in $\mathbb{F}_q^2$ is negligible, namely at most $\frac{2}{\sqrt{p}}$.[13] In this context, observe that we are actually not forced to do ElGamal encryption in the pairing group that BBS uses. We could alternatively use ElGamal (or any other secure encryption) over any other group as long as we are able find a suitable pseudorandom encoding of a ciphertext as $\mathbb{Z}_q^\mu$ element.

The 4-message anamorphic extension $\Pi_{\mathsf{aBBS}}$ resulting from a pseudorandom $(\mathbb{G}_1^2, \mathbb{Z}_q^4)$-encoding (e.g., using Elligator Squared for $\mathbb{G}_1$ of BN curves) is shown in Figure 9. Note that no trapdoor is required here to compute the second signature component. $\Pi_{\mathsf{aBBS}}$ achieves anamorphic security and IND-CPA security. To achieve sIND-RCCA security, it suffices to replace plain ElGamal by an FO-based PKE from ElGamal and XOR-based symmetric encryption and extend the pseudorandom encoding described above to also cover the symmetric ciphertext component. For $q$ close to $2^\lambda$, a random bitstring $G(r) \oplus \mathsf{amsg} \in \{0,1\}^\lambda$ can simply be interpreted as $\mathbb{F}_q$-element $s_5$ which causes an encoding error (integer $> p-1$) with negligible probability. This leads to a 5-message anamorphic signature extension satisfying sIND-RCCA security. To additionally achieve IND-CCA security, one needs to make use of the SUF-CMA security of BBS, by checking the validity of the signatures **asig** wrt. to the messages **msg** in aDec.

Also for other variants of BBS signatures like BBS+ [ASM06], such anamorphic extensions can be obtained in the same way. The security of all these extensions is covered by our master theorem in Section 6.1, which is discussed below.

**Theorem 9** *BBS signatures (Figure 9) are anamorphism-friendly and ElGamal encryption, where ciphertexts are encoded during encryption and decoded during decryption using a pseudorandom $\mathbb{Z}_p^k$-encoding, is compliant to BBS under the* DDH *assumption.*

**Proof.** We prove the theorem in two parts.

**Anamorphism-Friendliness.** The signature space of BBS can be written in the form of Definition 28 as $S_0 \times R$, where $S_0 = \mathbb{Z}_p$ and $R = \mathbb{G}$. As the first component of each signature $(s_0, H(\mathsf{msg})^{\frac{1}{x+s_0}})$ is uniformly and freshly chosen, the first property is satisfied. The second one is also trivially satisfied by setting $\mathsf{td} := \bot$ and $\mathsf{RemSig}(\mathsf{ask} = x, \mathsf{td}, \mathsf{msg}, s_0) := H(\mathsf{msg})^{\frac{1}{x+s_0}}$.

**Compliance of ElGamal+pseudorandom Encoding.** Let us consider ElGamal encryption over $\mathbb{G}_1$ of an asymmetric bilinear groups setting $\mathsf{gp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, , g_1, g_2, e, p)$ and Encode and Decode algorithms mapping elements from $\mathbb{G}_1$ to $\mathbb{Z}_p^k$ and vice versa. Then the encryption scheme defined by $\mathsf{Gen}(\mathsf{gp}) := \mathsf{El.Gen}(\mathsf{gp})$, $\mathsf{Enc}(\mathsf{pk}, \mathsf{msg}) := (\mathsf{Encode}(c_1), \mathsf{Encode}(c_2))$, where $(c_1, c_2) \leftarrow \mathsf{El.Enc}(\mathsf{pk}, \mathsf{msg})$, $\mathsf{Dec}(\mathsf{sk}, (c_1', c_2')) := \mathsf{El.Dec}(\mathsf{sk}, (\mathsf{Decode}(c_1'), \mathsf{Decode}(c_2')))$ has ciphertext space $\mathbb{Z}_p^{2k}$, provides correctness up to a negligible error probability, and satisfies IND$-CPA security. Regarding IND$-CPA security, we first observe that ElGamal is IND$-CPA secure under the DDH assumption (the proof is similar to the one for linear encryption under DLIN), and that the output of Encode for some uniform $c_i \in \mathbb{G}_1$ is indistinguishable from an uniform element

[13] To be compliant to the definition of our pseudorandom encoding, we could output $\bot$ in this case.

of $\mathbb{Z}_p^k$. Finally, observe that the ciphertext space of the combined encryption scheme is $\mathbb{Z}_p^{2k}$ and is thus compliant to the signature space component $S_0 = \mathbb{Z}_p$ of the BBS signature scheme. □

# 7 Anamorphism Notifier in Communication Streams

We present our protocols SCAN and SSAN for identifying anamorphic ciphertexts and signatures within communication streams. These protocols are designed to preserve the security guarantees of the underlying anamorphic encryption or signature scheme: the signaling mechanism, implemented via a pseudorandom function on prior ciphertexts or signatures, does not reveal information beyond what is already implicit in their distribution, except for the designated signaling object. For simplicity, we focus on CPA security; the same arguments extend to stronger notions such as CCA security.

## 7.1 Anamorphism Notifier for Encryption

In a general stream of ciphertexts, multiple anamorphic messages may be embedded at arbitrary positions. In such cases, SCAN will systematically scan through the ciphertext stream and efficiently identify all ciphertexts that contain anamorphic components, thereby enabling the successful recovery of every covert message. For the sake of clarity in exposition and to formalize the core construction, however, we consider a stream of fixed length $\theta$ in the definition of $\mathsf{SCAN}_\theta$ that contains exactly one anamorphic message embedded using an arbitrary anamorphic encryption scheme. The protocol samples a notification key, denoted as $\mathsf{nk}$, which is independent of other keys in the system. In general, the notification key can be sampled at any point and included in the double keys when anamorphic messages are embedded into the stream.

**Definition 29 (SCAN)** *A stream ciphertext anamorphism notifier (SCAN) with streaming length $\theta$ for a $\mu$-message AAEE scheme $\Pi_{\mathsf{AAE}} = (\mathsf{aGen}, \mathsf{aEnc}, \mathsf{aDec})$ with anamorphic message space $\widehat{\mathcal{M}}$ associated with a public key encryption scheme $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with a regular message space $\mathcal{M}$ is a sender-receiver protocol $\mathsf{SCAN}_\theta = (\mathsf{nGen}, \mathsf{Sen}, \mathsf{aSen}, \mathsf{Rec})$ which works as follows:*

$\mathsf{nk} \leftarrow \mathsf{nGen}(1^\lambda)$ : *The setup algorithm takes as input the security parameter $1^\lambda$, and outputs a notification key $\mathsf{nk}$ for signaling.*

$\mathbf{ct} \leftarrow \mathsf{Sen}(\mathsf{apk}, \mathbf{msg})$ : *The normal sender algorithm takes as input the public key $\mathsf{apk}$ and a sequence of normal messages $\mathbf{msg} = (\mathsf{msg}_1, \ldots, \mathsf{msg}_\theta) \in \mathcal{M}^\theta$, and outputs a sequence of ciphertexts $\mathbf{ct} \in \mathcal{CT}^\theta$ where $(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda)$.*

$\mathbf{act} \leftarrow \mathsf{aSen}(\mathsf{apk}, \mathsf{dk}, \mathsf{nk}, \mathbf{msg}, \mathsf{anam} = (s, \mathsf{amsg}))$ : *The anamorphic sender algorithm takes as input the public key $\mathsf{apk}$, the anamorphic public key $\mathsf{dk}$, the notification key $\mathsf{nk}$, a sequence of regular messages $\mathbf{msg} = (\mathsf{msg}_1, \ldots, \mathsf{msg}_\theta) \in \mathcal{M}^\theta$, an anamorphic message $\mathsf{anam}$ consisting of a signal $s \in [\theta - \mu]$ and an anamorphic message $\mathsf{amsg} \in \widehat{\mathcal{M}}$, and outputs a sequence of ciphertext $\mathbf{act} \in \mathcal{CT}^\theta$.*

$(\mathbf{msg}, \mathsf{anam}) \leftarrow \mathsf{Rec}(\mathsf{apk}, \mathsf{tk}, \mathsf{nk}, \mathbf{ct})$ : *The receiver algorithm takes as input the public key $\mathsf{apk}$, the anamorphic secret key $\mathsf{tk}$, the notification key $\mathsf{nk}$ and a sequence of ciphertexts $\mathbf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_\theta) \in \mathcal{CT}^\theta$, and outputs $(\mathbf{msg}, \mathsf{anam})$ or $\perp$.*

**Correctness for regular message**: *For every sequence of messages $\mathbf{msg} \in \mathcal{M}^\theta$, the following holds:*

$$\Pr\left[(\mathbf{msg}, \perp) = \mathsf{Rec}(\mathsf{apk}, \mathsf{tk}, \mathsf{nk}, \mathbf{ct}) : \begin{array}{c}(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda) \\ \mathsf{nk} \leftarrow \mathsf{nGen}(1^\lambda) \\ \mathbf{ct} \leftarrow \mathsf{Sen}(\mathsf{apk}, \mathbf{msg})\end{array}\right] = 1$$

*where the probability is taken over the random coin tosses of $\mathsf{aGen}$ and $\mathsf{Sen}$.*

**Correctness for anamorphic message**: *For every sequence of messages* $\mathbf{msg} \in \mathcal{M}^\theta$ *and* $\mathsf{amsg} \in \widehat{\mathcal{M}}$*, and for all* $s \in [\theta - \mu]$ *with* $\theta > \mu$*, the following holds:*

$$\Pr\left[(\mathbf{msg}, (s, \mathsf{amsg})) = \mathsf{Rec}(\mathsf{apk}, \mathsf{tk}, \mathsf{nk}, \mathbf{ct}) : \begin{array}{c}(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda) \\ \mathsf{nk} \leftarrow \mathsf{nGen}(1^\lambda) \\ \mathbf{ct} \leftarrow \mathsf{aSen}(\mathsf{apk}, \mathsf{dk}, \mathsf{nk}, \mathbf{msg}, \mathsf{anam})\end{array}\right] = 1$$

*where the probability is taken over the random coin tosses of* $\mathsf{nGen}, \mathsf{aGen}, \mathsf{aSen}$*.*

**Efficiency**: $\mathsf{SCAN}_\theta$ *can efficiently recover the anamorphic message, meaning that the number of times* aDec *is run by the* Rec *algorithm is independent of* $\theta$*. More formally,* $\mathsf{Time}_{\mathsf{Rec}} \approx \theta(\mathsf{Time}_{\mathsf{Enc}} + \mathsf{Time}_{\mathsf{Dec}}) + \mathsf{Time}_{\mathsf{aDec}}$ *where* $\mathsf{Time}_{\mathsf{X}}$ *denotes the runtime of the algorithm* X*.*

**Anamorphic security**: $\mathsf{SCAN}_\theta$ *satisfies anamorphic security if for every PPT dictator* $\mathcal{D}$ *it holds that*

$$\left|\Pr[\mathsf{GReal}^{\mathcal{D}}_{\mathsf{SCAN}}(\lambda) = 1] - \Pr[\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{SCAN}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security games are defined as follows:*

$\underline{\mathsf{GReal}^{\mathcal{D}}_{\mathsf{SCAN}}(\lambda):}$

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
2. ***return*** $\mathcal{D}^{\mathsf{OSen}(\cdot,\cdot)}(\mathsf{pk}, \mathsf{sk})$

*where* $\mathsf{OSen}(\mathbf{msg}, \mathsf{anam}) = \mathbf{ct}$*,*

$\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}, \mathsf{msg}_i)$*,*

$\mathbf{msg} = (\mathsf{msg}_i)_{i\in[\theta]}, \mathbf{ct} = (\mathsf{ct}_i)_{i\in[\theta]}$

$\underline{\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{SCAN}}(\lambda):}$

1. $(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda)$
2. $\mathsf{nk} \leftarrow \mathsf{nGen}(1^\lambda)$
3. ***return*** $\mathcal{D}^{\mathsf{OaSen}(\cdot,\cdot)}(\mathsf{apk}, \mathsf{ask})$

*where* $\mathsf{OaSen}(\mathbf{msg}, \mathsf{anam}) =$

$\mathsf{aSen}(\mathsf{apk}, \mathsf{dk}, \mathsf{nk}, \mathbf{msg}, \mathsf{anam})$

**Fully asymmetric IND-CPA security**: $\mathsf{SCAN}_\theta$ *satisfies fully asymmetric IND-CPA security if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all* $\lambda \in \mathbb{N}$*, the following holds:*

$$\left|\Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{SCAN}}(\lambda, 0) = 1] - \Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{SCAN}}(\lambda, 1) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security game is defined as follows:*

$\underline{\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{SCAN}}(\lambda, \beta):}$

1. $(\mathsf{apk}, \mathsf{ask}, \mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda)$
2. $\mathsf{nk} \leftarrow \mathsf{nGen}(1^\lambda)$
3. $(\mathbf{msg}_0, \mathbf{msg}_1, s, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk}, \mathsf{dk}, \mathsf{nk})$
4. $\mathbf{act}_\beta \leftarrow \mathsf{aSen}(\mathsf{apk}, \mathsf{dk}, \mathsf{nk}, \mathbf{msg}_\beta, \mathsf{anam} = (s, \mathsf{amsg}_\beta))$
5. ***return*** $\mathcal{A}(\mathbf{act}_\beta)$ *if* $\mathsf{msg}_{0,s} = \mathsf{msg}_{1,s}$ *where* $\mathbf{msg}_\beta = (\mathsf{msg}_{\beta,i})_{i\in[\theta]}$

We additionally consider strong IND-CPA security for $\mathsf{SCAN}_\theta$.

**Strong IND-CPA security**: $\mathsf{SCAN}_\theta$ satisfies strong IND-CPA (sIND-CPA) security if for every PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\left|\Pr[\mathsf{GsIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{SCAN}}(\lambda,0)=1] - \Pr[\mathsf{GsIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{SCAN}}(\lambda,1)=1]\right| \leq \mathsf{negl}(\lambda)$$

where the security game is defined as follows:

$\underline{\mathsf{GsIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{SCAN}}(\lambda,\beta):}$

1. $(\mathsf{apk},\mathsf{ask},\mathsf{dk},\mathsf{tk}) \leftarrow \mathsf{aGen}(1^\lambda)$
2. $\mathsf{nk} \leftarrow \mathsf{nGen}(1^\lambda)$
3. $(\mathbf{msg}, s, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{apk},\mathsf{ask},\mathsf{dk})$
4. $\mathbf{act}_\beta \leftarrow \mathsf{aSen}(\mathsf{apk},\mathsf{dk},\mathsf{nk},\mathbf{msg},\mathsf{anam}=(s,\mathsf{amsg}_\beta))$
5. **return** $\mathcal{A}(\mathbf{act}_\beta)$

Now, we formally describe our *fully generic* protocol $\mathsf{SCAN}_\theta$ in Figure 10 for any $\mu$-message AAE $\Pi_{\mathsf{AAE}} = (\mathsf{aGen},\mathsf{aEnc},\mathsf{aDec})$, with an associated streaming parameter $\theta$ such that $s+\mu \leq \theta$ where $s$ is the signal for the notification of anamorphism. Let $\mathcal{R}$ be the randomness space of the encryption algorithm and $\mathcal{CT}$ be the ciphertext space of the associated public key encryption scheme $\Pi_{\mathsf{PKE}} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$. The protocol requires a PRF $F : \mathcal{K} \times \{0,1\}^{|\mathsf{ct}|} \to \mathcal{R}$ where $|\mathsf{ct}|$ denotes the length of ciphertexts of the PKE.

**Correctness**: The correctness for regular message follows directly from the correctness of $\Pi_{\mathsf{PKE}}$ and the correctness of anamorphic message follows from that of $F$, $\Pi_{\mathsf{PKE}}$ and $\Pi_{\mathsf{AAE}}$.

**Efficiency**: By construction, Rec algorithm runs the aDec of $\Pi_{\mathsf{AAE}}$ only once. Therefore, we have $\mathsf{Time}_{\mathsf{Rec}} \approx \theta(\mathsf{Time}_{\mathsf{Enc}} + \mathsf{Time}_{\mathsf{Dec}}) + \mathsf{Time}_{\mathsf{aDec}}$.

**Theorem 10** *Assuming $F$ is a secure pseudorandom function and $\Pi_{\mathsf{AAE}}$ satisfies anamorphic, fully asymmetric IND-CPA security and sIND-CPA security, the stream ciphertext anamorphism notifier protocol $\mathsf{SCAN}_\theta$ described in Figure 10 satisfies anamorphic, fully asymmetric IND-CPA and sIND-CPA security.*

**Proof.** We prove the theorem in three parts.

**Anamorphic security.** We use four hybrid games Game 0, Game 1, Game 2 and Game 3 to prove the anamorphic security. Game 0 is exactly the same as $\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{SCAN}}(\lambda)$. The dictator gets oracle access to $\mathsf{OaSen}(\cdot,\cdot)$ and $\mathsf{ct}_s$ is computed as $\mathsf{Enc}(\mathsf{apk},\mathsf{msg}_s; F(K,\mathsf{ct}'))$ where $\mathsf{ct}' = 0^{|\mathsf{ct}|}$ if $s=1$ and $\mathsf{ct}' = \mathsf{ct}_{s-1}$ if $s>1$. In Game 1, the encryption randomness $F(K,\mathsf{ct}')$ is replaced with a uniform randomness. Since the dictator is not given the key $\mathsf{nk}=K$, Game 0 is indistinguishable from Game 1 by the security of $F$. In particular, distinguishing requires either breaking the PRF security of $F$, or hitting a collision with one of the at most $(\theta-1)$ other ciphertext randomnesses in the same stream; the latter occurs with negligible probability of at most $Q\cdot\theta/|\mathcal{R}|$ over all $Q$ oracle calls. Observe that Game 1 can be simulated without sampling the key $K$. In Game 2, we redefine $\mathsf{OaSen}(\mathbf{msg},\mathsf{anam})$ as follows: it computes $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{apk},\mathsf{msg}_i)$ for all $i \in [s] \cup [s+\mu+1,\theta]$ and $\mathsf{act}_\mu = \mathsf{aEnc}(\mathsf{apk},\mathsf{dk},\mathsf{msg}_\mu,\mathsf{amsg})$ where $\mathsf{msg}_\mu = (\mathsf{msg}_{s+1},\ldots,\mathsf{msg}_{s+\mu})$. It is easy to see that Game 2 is identical to Game 1. We define Game 3 as exactly the same as $\mathsf{GReal}^{\mathcal{D}}_{\mathsf{SCAN}}(\lambda)$ where $\mathsf{OaSen}(\cdot,\cdot)$ is replaced with $\mathsf{OSen}(\cdot,\cdot)$. Note that, Game 2 and Game 3 are indistinguishable under the anamorphic security of $\Pi_{\mathsf{AAE}}$.

**Fully asymmetric IND-CPA security.** It follows from a hybrid argument and relying on the fully asymmetric IND-CPA security of $\Pi_{\mathsf{AAE}}$ and IND-CPA security of $\Pi_{\mathsf{PKE}}$ under the key apk, since $\mathsf{msg}_{0,s} = \mathsf{msg}_{1,s}$.

$\underline{\mathsf{nGen}(1^\lambda):}$

1. $K \leftarrow \mathcal{K}$
2. $\mathsf{nk} \coloneqq K$
3. **return** $\mathsf{nk}$

$\underline{\mathsf{Rec}(\mathsf{tk}, \mathsf{nk}, \mathbf{ct}):}$

1. Parse $\mathbf{ct} = (\mathsf{ct}_i)_{i\in[\theta]}$
2. $s, \mathsf{amsg} \coloneqq \perp$
3. $\mathsf{msg}_i \coloneqq \mathsf{Dec}(\mathsf{ask}, \mathsf{ct}_i)\ \forall\ i \in [\theta]$
4. $\mathbf{msg} \coloneqq (\mathsf{msg}_i)_{i\in[\theta]}$
5. $s \coloneqq 1$ **if**
   $\mathsf{ct}_1 = \mathsf{Enc}(\mathsf{apk}, \mathsf{msg}_i; F(\mathsf{nk}, 0^{|\mathsf{ct}|}))$
6. $s \coloneqq i > 1$ **if**
   $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{apk}, \mathsf{msg}_i; F(\mathsf{nk}, \mathsf{ct}_{i-1}))$
7. **if** $s \neq \perp$,
   $\mathbf{act}_\mu \coloneqq (\mathsf{ct}_{s+1}, \ldots, \mathsf{ct}_{s+\mu})$,
   $\mathsf{amsg} \coloneqq \mathsf{aDec}(\mathsf{tk}, \mathbf{act}_\mu)$
8. **return** $(\mathbf{msg}, \mathsf{anam} \coloneqq (s, \mathsf{amsg}))$

$\underline{\mathsf{aSen}(\mathsf{apk}, \mathsf{dk}, \mathsf{nk}, \mathbf{msg}, \mathsf{anam} = (s, \mathsf{amsg})):}$

1. Parse $\mathbf{msg} = (\mathsf{msg}_i)_{i\in[\theta]}$
2. **if** $s = 1$,
   $\mathsf{ct}_s \coloneqq \mathsf{Enc}(\mathsf{apk}, \mathsf{msg}_s; F(\mathsf{nk}, 0^{|\mathsf{ct}|}))$
3. **if** $s > 1$,
   $$\mathsf{ct}_i \coloneqq \begin{cases} \mathsf{Enc}(\mathsf{apk}, \mathsf{msg}_i), & \forall\ i \in [s-1] \\ \mathsf{Enc}(\mathsf{apk}, \mathsf{msg}_i; F(\mathsf{nk}, \mathsf{ct}_{i-1})), & \textbf{if } i = s \end{cases}$$
4. $\mathbf{msg}_\mu \coloneqq (\mathsf{msg}_{s+1}, \ldots, \mathsf{msg}_{s+\mu})$
5. $\mathbf{act}_\mu \coloneqq \mathsf{aEnc}(\mathsf{apk}, \mathsf{dk}, \mathbf{msg}_\mu, \mathsf{amsg})$
6. $\mathsf{ct}_i \coloneqq \mathsf{Enc}(\mathsf{apk}, \mathsf{msg}_i)\ \forall\ i \in [s+\mu+1, \theta]$
7. **return**
   $\mathbf{act} \coloneqq (\mathsf{ct}_1, \ldots, \mathsf{ct}_s, \mathbf{act}_\mu, \mathsf{ct}_{s+\mu+1}, \ldots, \mathsf{ct}_\theta)$

$\underline{\mathsf{Sen}(\mathsf{apk}, \mathbf{msg}):}$

1. Parse $\mathbf{msg} = (\mathsf{msg}_i)_{i\in[\theta]}$
2. $\mathsf{ct}_i \coloneqq \mathsf{Enc}(\mathsf{apk}, \mathsf{msg}_i)\ \forall\ i \in [\theta]$
3. **return** $\mathbf{ct} \coloneqq (\mathsf{ct}_1, \ldots, \mathsf{ct}_\theta)$

Figure 10: $\mathsf{SCAN}_\theta$ for $\mu$-message $\Pi_{\mathsf{AAE}}$

**sIND-CPA security.** It follows from the sIND-CPA security of $\Pi_{\mathsf{AAE}}$. □

## 7.2 Anamorphism Notifier for Signatures

We now introduce our stream signature anamorphism notifier (SSAN), a framework that is structurally akin to SCAN, yet differs in subtle but significant ways. To begin, our generic signaling technique for identifying anamorphic messages in a stream of signatures only holds for a $\mu$-message AASE that satisfies Definition 28. Unlike the Rec protocol in SCAN, where a receiver derives randomness $r_k \leftarrow F(\mathsf{nk}, \mathsf{ct}_{k-1})$, and re-encrypts message $\mathsf{msg}_k$ using apk to check if it matches $\mathsf{ct}_k$ — a receiver in SSAN cannot recompute a signature $\mathsf{sig}_k$ since it does not know the sender's anamorphic signing key ask. We get around this apparent constraint by allowing the SSAN receiver to verify if the first $(m-1)$ uniformly random components of $\mathsf{sig}_k$ are indeed derived using $F(\mathsf{nk}, \mathsf{sig}_{k-1})$, which is feasible given the anamorphic receiver knows the signaling notification key nk. As in SCAN, the SSAN protocol samples nk independently of the other keys.

**Definition 30 (SSAN)** *A stream signature anamorphism notifier (SSAN) with streaming length $\theta$ for a $\mu$-message AASE scheme $\Pi_{\mathsf{AASE}} = (\mathsf{aGenS}, \mathsf{aGenR}, \mathsf{aSig}, \mathsf{aDec})$ with anamorphic message space $\widehat{\mathcal{M}}$ associated with a signature scheme $\Pi_{\mathsf{SS}} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Verify})$ with a regular message space $\mathcal{M}$, where $\mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda)$, is a sender-receiver protocol $\mathsf{SSAN}_\theta = (\mathsf{nGen}, \mathsf{Sen}, \mathsf{aSen}, \mathsf{Rec})$ which works as follows:*

$\underline{\mathsf{nk} \leftarrow \mathsf{nGen}(1^\lambda)}$ : *The setup algorithm takes as input the security parameter* $1^\lambda$*, and outputs a notification key* $\mathsf{nk}$ *for signalling.*

$\underline{\mathbf{sig} \leftarrow \mathsf{Sen}(\mathsf{ask}, \mathbf{msg})}$ : *The regular sender algorithm takes as input the sender signing key* $\mathsf{ask}$ *and a sequence of regular messages* $\mathbf{msg} = (\mathsf{msg}_1, \ldots, \mathsf{msg}_\theta) \in \mathcal{M}^\theta$*, and outputs a sequence of signatures* $\mathbf{sig} \in \mathcal{SIG}^\theta$*, where* $(\mathsf{ask}, \mathsf{avk}, \mathsf{td}) \leftarrow \mathsf{aGenS}(\mathsf{gp})$.

$\underline{\mathbf{asig} \leftarrow \mathsf{aSen}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathsf{nk}, \mathbf{msg}, \mathsf{anam} = (s, \mathsf{amsg}))}$ : *The anamorphic sender algorithm takes as input the sender signing key* $\mathsf{ask}$ *with associated trapdoor information* $\mathsf{td}$*, the receiver's encryption double key* $\mathsf{dk}$*, the notification key* $\mathsf{nk}$*, a sequence of regular messages* $\mathbf{msg} = (\mathsf{msg}_1, \ldots, \mathsf{msg}_\theta) \in \mathcal{M}^\theta$*, an anamorphic message* $\mathsf{anam}$ *consisting of a signal* $s \in [\theta - \mu]$ *and an anamorphic message* $\mathsf{amsg} \in \widehat{\mathcal{M}}$*, and outputs a sequence of signatures* $\mathbf{asig} \in \mathcal{SIG}^\theta$*, where* $(\mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGenR}(\mathsf{gp})$.

$\underline{(\mathbf{b}, \mathsf{anam}) \leftarrow \mathsf{Rec}(\mathsf{avk}, \mathsf{tk}, \mathsf{nk}, \mathbf{msg}, \mathbf{sig})}$ : *The receiver algorithm takes as input the sender's signature verification key* $\mathsf{avk}$*, the receiver's secret decryption double key* $\mathsf{tk}$*, the notification key* $\mathsf{nk}$ *and a sequence of signatures* $\mathbf{sig} = (\mathsf{sig}_1, \ldots, \mathsf{sig}_\theta) \in \mathcal{SIG}^\theta$ *along with a message vector* $\mathbf{msg}$*, and outputs* $(\mathbf{b}, \mathsf{anam})$*, where* $\mathbf{b}$ *is a bit string of length* $\theta$.

**Correctness for regular message**: *For every sequence of messages* $\mathbf{msg} \in \mathcal{M}^\theta$*, the following holds:*

$$\Pr\left[\begin{array}{c}(\mathbf{b}, \perp) = \mathsf{Rec}(\mathsf{avk}, \mathsf{tk}, \mathsf{nk}, \mathbf{msg}, \mathbf{sig}) \\ \textit{where } b_i \leftarrow \mathsf{Verify}(\mathsf{avk}, \mathsf{msg}_i, \mathsf{sig}_i) \\ \textit{and } \mathbf{b} = (b_1, \ldots, b_\theta)\end{array} : \begin{array}{c}(\mathsf{ask}, \mathsf{avk}, \mathsf{td}) \leftarrow \mathsf{aGenS}(\mathsf{gp}) \\ (\mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGenR}(\mathsf{gp}) \\ \mathsf{nk} \leftarrow \mathsf{nGen}(1^\lambda) \\ \mathbf{sig} \leftarrow \mathsf{Sen}(\mathsf{ask}, \mathbf{msg})\end{array}\right] = 1$$

*where the probability is taken over the random coin tosses of* $\mathsf{aGenS}$*,* $\mathsf{aGenR}$ *and* $\mathsf{Sen}$.

**Correctness for anamorphic message**: *For every sequence of messages* $\mathbf{msg} \in \mathcal{M}^\theta$ *and* $\mathsf{amsg} \in \widehat{\mathcal{M}}$*, and for all* $s \in [\theta - \mu]$ *with* $\theta > \mu$*, the following holds:*

$$\Pr\left[\begin{array}{c}(\mathbf{b}, (s, \mathsf{amsg})) = \mathsf{Rec}(\mathsf{avk}, \mathsf{tk}, \mathsf{nk}, \mathbf{msg}, \mathbf{sig}) \\ \textit{where } b_i \leftarrow \mathsf{Verify}(\mathsf{avk}, \mathsf{msg}_i, \mathsf{sig}_i) \\ \textit{and } \mathbf{b} = (b_1, \ldots, b_\theta)\end{array} : \begin{array}{c}(\mathsf{ask}, \mathsf{avk}, \mathsf{td}) \leftarrow \mathsf{aGenS}(\mathsf{gp}) \\ (\mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGenR}(\mathsf{gp}) \\ \mathsf{nk} \leftarrow \mathsf{nGen}(1^\lambda) \\ \mathbf{sig} \leftarrow \mathsf{aSen}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathsf{nk}, \mathbf{msg}, \mathsf{anam})\end{array}\right] = 1$$

*where probability is taken over the random coin tosses of* $\mathsf{nGen}, \mathsf{aGenS}, \mathsf{aGenR}$ *and* $\mathsf{aSen}$.

**Efficiency**: $\mathsf{SSAN}_\theta$ *can efficiently recover the anamorphic message, meaning that the number of times* $\mathsf{aDec}$ *is run by the* $\mathsf{Rec}$ *algorithm is independent of* $\theta$*. More formally,* $\mathsf{Time}_{\mathsf{Rec}} \approx \theta \cdot \mathsf{Time}_{\mathsf{Verify}} + \mathsf{Time}_{\mathsf{aDec}}$.

**Anamorphic security**: $\mathsf{SSAN}_\theta$ *satisfies anamorphic security if for every PPT dictator* $\mathcal{D}$ *it holds that*

$$\left|\Pr[\mathsf{GReal}^{\mathcal{D}}_{\mathsf{SSAN}}(\lambda) = 1] - \Pr[\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{SSAN}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security games are defined as follows:*

$\underline{\mathsf{GReal}^{\mathcal{D}}_{\mathsf{SSAN}}(\lambda):}$

1. $\mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(\mathsf{gp})$
3. ***return*** $\mathcal{D}^{\mathsf{OSen}(\cdot,\cdot)}(\mathsf{sk}, \mathsf{vk})$

*where* $\mathsf{OSen}(\mathbf{msg}, \mathsf{anam}) := \mathbf{sig}$, $\mathsf{sig}_i = \mathsf{Sig}(\mathsf{sk}, \mathsf{msg}_i)$, $\mathbf{msg} = (\mathsf{msg}_i)_{i\in[\theta]}, \mathbf{sig} = (\mathsf{sig}_i)_{i\in[\theta]}$

$\underline{\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{SSAN}}(\lambda):}$

1. $\mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{ask}, \mathsf{avk}, \mathsf{td}) \leftarrow \mathsf{aGenS}(\mathsf{gp})$
3. $(\mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGenR}(\mathsf{gp})$
4. $\mathsf{nk} \leftarrow \mathsf{nGen}(1^\lambda)$
5. ***return*** $\mathcal{D}^{\mathsf{OaSen}(\cdot,\cdot)}(\mathsf{ask}, \mathsf{avk})$

*where* $\mathsf{OaSen}(\mathbf{msg}, \mathsf{anam}) = \mathsf{aSen}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathsf{nk}, \mathbf{msg}, \mathsf{anam})$

**IND-CPA security**: $\mathsf{SSAN}_\theta$ *satisfies fully asymmetric IND-CPA security if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all* $\lambda \in \mathbb{N}$*, the following holds:*

$$\left|\Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{SSAN}}(\lambda, 0) = 1] - \Pr[\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{SSAN}}(\lambda, 1) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the security game is defined as follows:*

$\underline{\mathsf{GIND\text{-}CPA}^{\mathcal{A}}_{\mathsf{SSAN}}(\lambda, \beta):}$

1. $\mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{ask}, \mathsf{avk}, \mathsf{td}) \leftarrow \mathsf{aGenS}(\mathsf{gp})$
3. $(\mathsf{dk}, \mathsf{tk}) \leftarrow \mathsf{aGenR}(\mathsf{gp})$
4. $\mathsf{nk} \leftarrow \mathsf{nGen}(1^\lambda)$
5. $(\mathbf{msg}, s, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}(\mathsf{avk}, \mathsf{dk}, \mathsf{nk})$
6. $\mathbf{asig}_\beta \leftarrow \mathsf{aSen}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathsf{nk}, \mathbf{msg}, \mathsf{anam} = (s, \mathsf{amsg}_\beta))$
7. ***return*** $\beta' \leftarrow \mathcal{A}^{\mathsf{OaSen}(\cdot,\cdot)}(\mathbf{asig}_\beta)$ *where*

$\mathsf{OaSen}(\mathbf{msg}, \mathsf{amsg}) := \mathsf{aSen}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathsf{nk}, \mathbf{msg}, \mathsf{anam})$.

Now, we formally describe our *fully generic* protocol $\mathsf{SSAN}_\theta$ in Figure 11 for a $\mu$-message AASE $\Pi_{\mathsf{AASE}} = (\mathsf{aGenS}, \mathsf{aGenR}, \mathsf{aSig}, \mathsf{aDec})$ which is obtained from a signature scheme $\Pi_{\mathsf{SS}} = (\mathsf{Setup}, \mathsf{Gen}', \mathsf{Sig}, \mathsf{RemSig}, \mathsf{Verify})$ and an encryption scheme $\Pi_{\mathsf{ES}}$ compliant to $\Pi_{\mathsf{SS}}$ according to Definition 28. As before, we consider an associated streaming parameter $\theta$, such that $s+\mu \leq \theta$ where $s$ is the signal for the notification of anamorphism. Let $\Pi_{\mathsf{SS}}$ be a signature scheme with a signature space that can be written as $S_0^{e_0} \times \ldots \times S_{m-1}^{e_{m-1}} \times R$, with $S_i \neq S_j$, $e_i > 0$, $m \geq 0$. Then, a signature for a message $\mathsf{msg}$ takes the form $\mathsf{sig}_j = (\sigma_j = (\sigma_{0,j}, \ldots, \sigma_{m-1,j}), r_j)$, where all $\sigma_{i,j} \leftarrow S_i^{e_i}$ are uniformly random, and $r_j = \mathsf{RemSig}(\mathsf{sk}, \mathsf{td}, \mathsf{msg}, \sigma_j)$ is the remainder component of the signature. The $\mathsf{SSAN}$ protocol requires a PRF $F : \mathcal{K} \times \{0,1\}^{|\mathsf{sig}|} \to S_0^{e_0} \times \cdots \times S_{m-1}^{e_{m-1}}$, where $|\mathsf{sig}|$ denotes the length of signatures of $\Pi_{\mathsf{SS}}$.

**Correctness**: The correctness for regular message follows directly from the correctness of $\Pi_{\mathsf{SS}}$, and the correctness for anamorphic message follows from that of $F$ and $\Pi_{\mathsf{AASE}}$.

$\underline{\mathsf{nGen}(1^\lambda):}$

1. $K \leftarrow \mathcal{K}$
2. $\mathsf{nk} \coloneqq K$
3. **return** $\mathsf{nk}$

$\underline{\mathsf{Rec}(\mathsf{tk}, \mathsf{avk}, \mathsf{nk}, \mathbf{sig}, \mathbf{msg}):}$

1. Parse $\mathbf{sig} = (\mathsf{sig}_i = (\sigma_i, r_i))_{i\in[\theta]}$
2. Parse $\mathbf{msg} = (\mathsf{msg}_i)_{i\in[\theta]}$
3. $s, \mathsf{amsg} \coloneqq \perp$
4. $b_i \coloneqq \mathsf{Verify}(\mathsf{avk}, \mathsf{msg}_i, \mathsf{sig}_i)\ \forall i \in [\theta]$

   $\mathbf{b} \coloneqq (b_1, \ldots, b_\theta)$
5. $s \coloneqq 1$ **if**

   $\sigma_1 = F(\mathsf{nk}, 0^{|\mathsf{sig}|})$
6. $s \coloneqq i > 1$ **if**

   $\sigma_i = F(\mathsf{nk}, \mathsf{sig}_{i-1})$
7. **if** $s \neq \perp$,

   $\mathbf{asig}_\mu \coloneqq (\mathsf{sig}_{s+1}, \ldots, \mathsf{sig}_{s+\mu})$

   $\mathsf{amsg} \coloneqq \mathsf{aDec}(\mathsf{tk}, \mathbf{asig}_\mu)$
8. **return** $(\mathbf{b}, \mathsf{anam} \coloneqq (s, \mathsf{amsg}))$

$\underline{\mathsf{aSen}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathsf{nk}, \mathbf{msg}, \mathsf{anam} = (s, \mathsf{amsg})):}$

1. Parse $\mathbf{msg} = (\mathsf{msg}_i)_{i\in[\theta]}$
2. **if** $s = 1$,
3. $\sigma_1 \coloneqq F(\mathsf{nk}, 0^{|\mathsf{sig}|})$

   $r_1 \coloneqq \mathsf{RemSig}(\mathsf{ask}, \mathsf{td}, \mathsf{msg}_s, \sigma_1)$

   $\mathsf{sig}_1 \coloneqq (\sigma_1, r_1)$
4. **if** $s > 1$,

   $\sigma_s \coloneqq F(\mathsf{nk}, \mathsf{sig}_{s-1})$

   $r_s \coloneqq \mathsf{RemSig}(\mathsf{ask}, \mathsf{td}, \mathsf{msg}_s, \sigma_s)$

   $\mathsf{sig}_i \coloneqq \begin{cases} \mathsf{Sig}(\mathsf{ask}, \mathsf{msg}_i), & \forall\ i \in [s-1] \\ (\sigma_s, r_s) \end{cases}$
5. $\mathbf{msg}_\mu \coloneqq (\mathsf{msg}_{s+1}, \ldots, \mathsf{msg}_{s+\mu})$
6. $\mathbf{asig}_\mu \coloneqq \mathsf{aSig}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathbf{msg}_\mu, \mathsf{amsg})$
7. $\mathsf{sig}_i \coloneqq \mathsf{Sig}(\mathsf{ask}, \mathsf{msg}_i)\ \forall\ i \in [s+\mu+1, \theta]$
8. **return**

   $\mathbf{asig} \coloneqq (\mathsf{sig}_1, \ldots, \mathsf{sig}_s, \mathbf{asig}_\mu, \mathsf{sig}_{s+\mu+1}, \ldots, \mathsf{sig}_\theta)$

$\underline{\mathsf{Sen}(\mathsf{ask}, \mathbf{msg}):}$

1. Parse $\mathbf{msg} = (\mathsf{msg}_i)_{i\in[\theta]}$
2. $\mathsf{sig}_i \coloneqq \mathsf{Sig}(\mathsf{ask}, \mathsf{msg}_i)\ \forall\ i \in [\theta]$
3. **return** $\mathbf{sig} \coloneqq (\mathsf{sig}_1, \ldots, \mathsf{sig}_\theta)$

Figure 11: $\mathsf{SSAN}_\theta$ for $\mu$-message $\Pi_{\mathsf{AASE}}$

**Efficiency**: By construction, Rec algorithm runs the aDec of $\Pi_{\mathsf{AASE}}$ only once. Therefore, we have $\mathsf{Time}_{\mathsf{Rec}} \approx \theta \cdot \mathsf{Time}_{\mathsf{Verify}} + \mathsf{Time}_{\mathsf{aDec}}$.

**Theorem 11** *Assuming $F$ is a secure pseudorandom function and $\Pi_{\mathsf{AASE}}$ satisfies anamorphic and IND-CPA security, the stream signature anamorphism notifier protocol $\mathsf{SSAN}_\theta$ described in Figure 11 satisfies anamorphic and IND-CPA security.*

**Proof.** We prove the theorem in two parts.

**Anamorphic security.** We use four hybrid games Game 0, Game 1, Game 2 and Game 3 to prove the anamorphic security. Game 0 is exactly the same as $\mathsf{GAnam}^{\mathcal{D}}_{\mathsf{SSAN}}(\lambda)$. The dictator gets oracle access to $\mathsf{OaSen}(\cdot,\cdot)$ and $\mathsf{sig}_s$ is computed as $\mathsf{aSen}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathsf{nk}, \mathbf{msg}, \mathsf{anam}; F(K, \mathsf{sig}'))$ where $\mathsf{sig}' = 0^{|\mathsf{sig}|}$ if $s = 1$ and $\mathsf{sig}' = \mathsf{sig}_{s-1}$ if $s > 1$. In Game 1, the signature randomness $F(K, \mathsf{sig}')$ is replaced with uniform randomness. Since the dictator is not given the key $\mathsf{nk} = K$, Game 0 is indistinguishable from Game 1 by the security of

$F$. In particular, distinguishing requires either breaking the PRF security of $F$, or hitting a collision between the freshly sampled signaling randomness and one of the at most $(\theta - 1)$ other signature randomnesses in the same stream; the latter occurs with a negligible probability of at most $Q \cdot \theta/(|S_0|^{e_0} \cdots |S_{m-1}|^{e_{m-1}})$ over all $Q$ oracle calls.

Observe that $\mathsf{Game}\ 1$ can be simulated without sampling the key $K$. In $\mathsf{Game}\ 2$, we redefine $\mathsf{OaSen}(\mathbf{msg}, \mathsf{anam})$ as follows: it computes $\mathsf{sig}_i = \mathsf{Sig}(\mathsf{ask}, \mathsf{msg}_i)$ for all $i \in [s] \cup [s+\mu+1, \theta]$ and $\mathbf{asig}_\mu = \mathsf{aSig}(\mathsf{ask}, \mathsf{td}, \mathsf{dk}, \mathbf{msg}_\mu, \mathsf{amsg})$ where $\mathbf{msg}_\mu = (\mathsf{msg}_{s+1}, \ldots, \mathsf{msg}_{s+\mu})$. It is easy to see that $\mathsf{Game}\ 2$ is identical to $\mathsf{Game}\ 1$. We define $\mathsf{Game}\ 3$ as exactly the same as $\mathsf{GReal}^{\mathcal{D}}_{\mathsf{SSAN}}(\lambda)$ where $\mathsf{OaSen}(\cdot,\cdot)$ is replaced with $\mathsf{OSen}(\cdot,\cdot)$. Note that $\mathsf{Game}\ 2$ and $\mathsf{Game}\ 3$ are indistinguishable under the anamorphic security of $\Pi_{\mathsf{AASE}}$.

**IND-CPA security.** It follows from the IND-CPA security of $\Pi_{\mathsf{AASE}}$. □

**Acknowledgement.** The first three authors' work was funded by the Topic Engineering Secure Systems of the Helmholtz Association (HGF) and supported by KASTEL Security Research Labs, Karlsruhe. We thank the anonymous reviewers for their helpful comments.

# References

[ABG+25] Gennaro Avitabile, Vincenzo Botta, Emanuele Giunta, Marcin Mielniczuk, and Francesco Migliaro. The malice of ELFs: Practical anamorphic-resistant encryption without random oracles. Cryptology ePrint Archive, Paper 2025/305, 2025.

[ABR99] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. Dhaes: An encryption scheme based on the diffie-hellman problem. *IACR Cryptol. ePrint Arch.*, 1999:7, 1999.

[ADKL19] Prabhanjan Ananth, Apoorvaa Deshpande, Yael Tauman Kalai, and Anna Lysyanskaya. Fully homomorphic NIZK and NIWI proofs. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 356–385. Springer, Cham, December 2019.

[ASM06] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In Roberto De Prisco and Moti Yung, editors, *SCN 06: 5th International Conference on Security in Communication Networks*, volume 4116 of *Lecture Notes in Computer Science*, pages 111–125. Springer, Berlin, Heidelberg, September 2006.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, Berlin, Heidelberg, August 2004.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, Berlin, Heidelberg, August 2001.

[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, Part II*, volume 14652 of *Lecture Notes in Computer Science*, pages 3–32. Springer, Cham, May 2024.

[BN06] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *SAC 2005: 12th Annual International Workshop on Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, Berlin, Heidelberg, August 2006.

[Bon98] Dan Boneh. The decision diffie-hellman problem. In *International algorithmic number theory symposium*, pages 48–63. Springer, 1998.

[BR95] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, Berlin, Heidelberg, May 1995.

[Bra89] Robert Braden. Requirements for internet hosts – communication layers. Technical Report RFC 1122, Internet Engineering Task Force, 1989.

[BSW06] Dan Boneh, Emily Shen, and Brent Waters. Strongly unforgeable signatures based on computational Diffie-Hellman. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 229–240. Springer, Berlin, Heidelberg, April 2006.

[CC05] Liqun Chen and Zhaohui Cheng. Security proof of sakai-kasahara's identity-based encryption scheme. In Nigel P. Smart, editor, *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *Lecture Notes in Computer Science*, pages 442–459. Springer, 2005.

[CCLZ25] Wonseok Choi, Daniel Collins, Xiangyu Liu, and Vassilis Zikas. A unified treatment of anamorphic encryption. Cryptology ePrint Archive, Paper 2025/309, 2025.

[CDHT25] Hien Chu, Khue Do, Lucjan Hanzlik, and Sri AravindaKrishnan Thyagarajan. When threshold meets anamorphic signatures: What is possible and what is not! Cryptology ePrint Archive, Paper 2025/1044, 2025.

[CGM24a] 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, Part II*, volume 14652 of *Lecture Notes in Computer Science*, pages 33–62. Springer, Cham, May 2024.

[CGM24b] Dario Catalano, Emanuele Giunta, and Francesco Migliaro. Limits of black-box anamorphic encryption. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part II*, volume 14921 of *Lecture Notes in Computer Science*, pages 352–383. Springer, Cham, August 2024.

[CGM25] Dario Catalano, Emanuele Giunta, and Francesco Migliaro. Generic anamorphic encryption, revisited: New limitations and constructions. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part II*, volume 15602 of *Lecture Notes in Computer Science*, pages 275–303. Springer, Cham, May 2025.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582. Springer, Berlin, Heidelberg, August 2003.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, Berlin, Heidelberg, August 1998.

[DG25] Yevgeniy Dodis and Eli Goldin. Anamorphic-resistant encryption; or why the encryption debate is still alive. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part III*, volume 16002 of *Lecture Notes in Computer Science*, pages 440–471. Springer, Cham, August 2025.

[DH21] Whitfield Diffie and Martin Hellman. New directions in cryptography (1976). 2021.

[DL25a] Amit Deo and Benoît Libert. Anamorphic signatures with dictator and recipient unforgeability for long messages. In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology – ASIACRYPT 2025, Part VI*, volume 16250 of *Lecture Notes in Computer Science*, pages 370–401. Springer, Singapore, December 2025.

[DL25b] Amit Deo and Benoît Libert. Fully asymmetric anamorphic homomorphic encryption from LWE. 2025.

[DPPY26] Xuan Thanh Do, Giuseppe Persiano, Duong Hieu Phan, and Moti Yung. Randomness-recovery trapdoors: a new methodology for enhancing anamorphic encryption. Cryptology ePrint Archive, Paper 2026/135, 2026.

[DR08] Tim Dierks and Eric Rescorla. The transport layer security (tls) protocol version 1.2. Technical Report RFC 5246, Internet Engineering Task Force, 2008.

[ElG86] Taher ElGamal. On computing logarithms over finite fields. In Hugh C. Williams, editor, *Advances in Cryptology – CRYPTO'85*, volume 218 of *Lecture Notes in Computer Science*, pages 396–402. Springer, Berlin, Heidelberg, August 1986.

[ETS22] ETSI. Etsi tr 103 719: Guide to identity-based cryptography. https://www.etsi.org/deliver/etsi_tr/103700_103799/103719/01.01.01_60/tr_103719v010101p.pdf, 2022.

[FHKP13] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 254–271. Springer, Berlin, Heidelberg, February / March 2013.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013.

[Gal05] David Galindo. Boneh-Franklin identity based encryption revisited. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005: 32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 791–802. Springer, Berlin, Heidelberg, July 2005.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC 2008*, pages 197–206. ACM, 2008.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, Berlin, Heidelberg, August 2013.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st Annual ACM Symposium on Theory of Computing*, pages 12–24. ACM Press, May 1989.

[IT21] Jana Iyengar and Martin Thomson. Quic: A udp-based multiplexed and secure transport. Technical Report RFC 9000, Internet Engineering Task Force, 2021.

[JS24] Joseph Jaeger and Roy Stracovsky. Dictators? Friends? Forgers. - breaking and fixing unforgeability definitions for anamorphic signature schemes. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part II*, volume 15485 of *Lecture Notes in Computer Science*, pages 105–137. Springer, Singapore, December 2024.

[KLR19] Michael Klooß, Anja Lehmann, and Andy Rupp. (R)CCA secure updatable encryption with integrity protection. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 68–99. Springer, Cham, May 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, *Advances in Cryptology – CRYPTO 2023, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 759–790. Springer, Cham, 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. *Proceedings on Privacy Enhancing Technologies*, 2023(4):170–183, 2023.

[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, *Advances in Cryptology – CRYPTO 2019, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 671–700. Springer, Cham, August 2019.

[KYH+06] Takashi Kitagawa, Peng Yang, Goichiro Hanaoka, Rui Zhang, Hajime Watanabe, Kanta Matsuura, and Hideki Imai. Generic transforms to acquire cca-security for identity based encryption: The cases of fopkc and REACT. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *Information Security and Privacy, 11th Australasian Conference, ACISP 2006, Melbourne, Australia, July 3-5, 2006, Proceedings*, volume 4058 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2006.

[LKWL24] Tobias Looker, Vasilis Kalos, Andrew Whitehead, and Mike Lodder. The BBS Signature Scheme. Internet-Draft draft-irtf-cfrg-bbs-signatures-07, Internet Engineering Task Force, September 2024. Work in Progress.

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

[Pos81] Jon Postel. Transmission control protocol. Technical Report RFC 793, Internet Engineering Task Force, 1981.

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

[PPY24a] Giuseppe Persiano, Duong Hieu Phan, and Moti Yung. Public-key anamorphism in (CCA-secure) public-key encryption and beyond. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part II*, volume 14921 of *Lecture Notes in Computer Science*, pages 422–455. Springer, Cham, August 2024.

[PPY24b] Giuseppe Persiano, Duong Hieu Phan, and Moti Yung. Public-key anamorphism in (CCA-secure) public-key encryption and beyond. Cryptology ePrint Archive, Paper 2024/1327, 2024.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[Res18] Eric Rescorla. The transport layer security (tls) protocol version 1.3. Technical Report RFC 8446, Internet Engineering Task Force, 2018.

[SAB+22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[SOK01] Ryuichi Sakai, Shigeo Ohgishi, and Masao Kasahara. Cryptosystems based on pairing over elliptic curve (in japanese). In Nigel P. Smart, editor, *The 2001 Symposium on Cryptography and Information Security*, Lecture Notes in Computer Science, 2001.

[Tib14] Mehdi Tibouchi. Elligator squared: Uniform points on elliptic curves of prime order as uniform random strings. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *FC 2014: 18th International Conference on Financial Cryptography and Data Security*, volume 8437 of *Lecture Notes in Computer Science*, pages 139–156. Springer, Berlin, Heidelberg, March 2014.

[TZ23] Stefano Tessaro and Chenzhi Zhu. Revisiting BBS signatures. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 691–721. Springer, Cham, April 2023.

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, Berlin, Heidelberg, May 2005.

[WCHY23] Yi Wang, Rongmao Chen, Xinyi Huang, and Moti Yung. Sender-anamorphic encryption reformulated: Achieving robust and generic constructions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 135–167. Springer, 2023.

[WHL25] Weihao Wang, Shuai Han, and Shengli Liu. Anamorphic authenticated key exchange: Double key distribution under surveillance. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 168–200. Springer, 2025.

[YKH+06] Peng Yang, Takashi Kitagawa, Goichiro Hanaoka, Rui Zhang, Kanta Matsuura, and Hideki Imai. Applying fujisaki-okamoto to identity-based encryption. In Marc P. C. Fossorier, Hideki Imai, Shu Lin, and Alain Poli, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 16th International Symposium, AAECC-16, Las Vegas, NV, USA, February 20-24, 2006, Proceedings*, volume 3857 of *Lecture Notes in Computer Science*, pages 183–192. Springer, 2006.

# A Informal: Robustness of Asymmetric Anamorphic Signature Extension

For an AASE or more precisely for anamorphism-friendly signatures with compliant encryption we could define robustness analogous to [BGH+24]. This captures that it must be hard to find $\mu$ messages $(\mathsf{msg}_1, \ldots, \mathsf{msg}_\mu)$ for which when generating a vector of normal signatures viewed as an anamorphic signature $\mathsf{asig} = (\mathsf{SS.Sig}(\mathsf{ask}, \mathsf{msg}_1), \ldots, \mathsf{SS.Sig}(\mathsf{ask}, \mathsf{msg}_\mu))$, which is then subsequently anamorphically decrypted into $\mathsf{amsg} := \mathsf{aDec}(\mathsf{tk}, \mathsf{asig})$, it holds that $\mathsf{amsg} \neq \bot$. Formally, as done for an anamorphic extension of a PKE in [BGH+24], this can be done by defining a distinguishing problem between an oracle implementing the aforementioned and an oracle that always simply returns $\bot$.

Unfortunately, it is not clear how robustness could generically be related to the IND$-CPA security of the compliant PKE scheme used to construct the AASE according to Definiton 28. When looking at our concrete constructions, however, we can observe the following.

It is easy to see that the 3-AASE $\Pi_{\mathsf{aWaters}}$ in Figure 8 uses an ElGamal type encryption (namely linear encryption) and thus we can follow the ElGamal asymmetric anamorphic 2-message extension $\Pi_{\mathsf{aEl}}$ in Figure

1. Namely, we could use linear encryption in the exponent, i.e., instead of encrypting $\mathsf{amsg} \in \mathbb{G}$ we encrypt $g^{\mathsf{amsg}}$ for $\mathsf{amsg}$ coming from a polynomially bounded subset of $\mathbb{Z}_p$, and follow the exact same idea.

For the 4-AASE $\Pi_{\mathsf{aBBS}}$ in 9, although using ElGamal encryption and thus suggesting the use of the same strategy as above, it is not obvious how to achieve this. More precisely, when we consider a vector of $\mu$ normal signatures, then elements $s_i, s_j$, $i \neq j$ from any such two signatures when decoded, need to yield a random element in $\mathbb{G}_1$. However, this depends on the concrete encoding and is something that would need to be satisfied by the Elligator Squared encoding in our specific case or in general by the algorithm $\mathsf{Decode}$ of the pseudo-random $\mathbb{Z}_p^k$-encoding.

We leave a thorough study of robustness in context of AASE and anamorphism-friendly signatures with compliant encryption for future work.