A preliminary version of this work appears in the Asiacrypt 2024 proceedings (©IACR 2024).
This is the full version.

Dictators? Friends? Forgers.

Breaking and Fixing Unforgeability Definitions for Anamorphic Signature Schemes

Joseph Jaeger and Roy Stracovsky

School of Cybersecurity and Privacy
Georgia Institute of Technology, Atlanta, Georgia, USA
`{josephjaeger, rstracovsky3}@gatech.edu`

November 19, 2025

Abstract. Anamorphic signature schemes (KPPYZ, Crypto 2023) allow users to hide encrypted messages in signatures to allow covert communication in a hypothesized scenario where encryption is outlawed by a "dictator" but authentication is permitted. We enhance the security of anamorphic signatures by proposing two parallel notions of unforgeability which close gaps in existing security definitions. The first notion considers a *dictator* who wishes to forge anamorphic signatures. This notion patches a divide between the definition and a stated security goal of robustness (BGHMR, Eurocrypt 2024). We port two related BGHMR constructions to the signature scheme setting and demonstrate that, as presented, both of these and a construction from KPPYZ are insecure under an active dictator. However, two of the three can easily be modified to satisfy our definition. The second notion we propose considers a *recipient* who wishes to forge signatures. To motivate this notion, we identify a gap in an existing security definition from KPPYZ and present attacks that allow parties to be impersonated when using schemes erroneously deemed secure. We then formalize our new unforgeability definition to close this gap. Interestingly, while the new definition is only modestly different from the old one, the change introduces subtle technical challenges that arise when proving security. We overcome these challenges in our reanalysis of existing anamorphic signature schemes by showing they achieve our new notion when built from chosen-randomness secure signatures or with encryption that satisfies a novel ideal-model simulatability property.

Contents

1 Introduction

Cryptography provides a diverse set of tools and techniques that afford privacy, confidentiality, authenticity, and anonymity, among a myriad of other goals. Each goal can foster people's security, allowing them to focus on other needs and ultimately empowering them. This empowerment protects people from those who wish to violate their security, which can include governments. A controlling government may thus want to screen, monitor, or restrict the use of cryptography.

Persiano, Phan, and Yung [PPY22] recently introduced a theoretical technique called anamorphic cryptography aimed at subverting or even dissuading such government control. Anamorphic cryptography envisions the dire setting of parties under the domain of a dictator who permits only limited forms of cryptography and can monitor all communication or force parties to give up their cryptographic keys in order to check for compliance. The authors then provided techniques for covert communication by hiding secret ciphertexts inside of other honestly generated ciphertexts. This work was followed up by [BGH$^+$24,CGM24,KPP$^+$23a,KPP$^+$23b,WCHY23] which refine and extend the original idea. Of note is [BGH$^+$24] by Banfi, Gegier, Hirt, Maurer, and Rito (BGHMR), which proposes a notion of robustness (and many constructions achieving it) that allows parties using anamorphic cryptography to determine whether or not the outputs of a cryptosystem contain covert messages.

We focus on the notion of anamorphic signature schemes introduced by Kutylowski, Persiano, Phan, Yung, and Zawada (KPPYZ) [KPP$^+$23a]. They hypothesize a particularly restrictive setting in which the dictator bans encryption (or nullifies it by acting as a middlebox that decrypts and re-encrypts all communication) but allows authentication, again with the caveat that they may coerce parties to surrender their signing keys. The authors show that covert communication is still possible, this time via anamorphic signature schemes which hide encrypted messages inside of innocuous signatures. Additionally, the authors provide schemes where the parties communicating covertly fully trust each other and schemes where that trust is eroded.

In this work, we improve the security of anamorphic signatures by proposing two parallel unforgeability definitions that comprehensively capture the anamorphic setting. The first notion considers unforgeability from the perspective of a dictator who knows each party's signing keys, while the second notion considers unforgeability from the perspective of an untrusted recipient privy to the hidden communication channel. As motivation for each notion, we first revisit two previously proposed security definitions from BGHMR and KPPYZ and observe crucial mismatches between the mathematical formalization of security and the expected deployment scenario. These mismatches allow natural or even previously proposed schemes that provably fulfill the old security definitions but would be trivially insecure in practice due to attacks we identify. Both of our new security definitions mend these gaps, and we show (with new definitions and proofs) conditions in which anamorphic signature schemes satisfy our proposed notions.

Provable Stealthiness. From an operational perspective, anamorphic cryptography is another theoretical tool in the basket of stealthy communication techniques. It is not meant to replace these techniques, but instead add another medium within which users can communicate covertly. This expands the stealthy bandwidth available to parties. From a security perspective, however, anamorphic cryptography provides strong provable security guarantees. One can consider (as Persiano, Phan, and Yung did in [PPY22]) a simple stealthiness game in which a "dictator" (or simply any adversarial observer) is given access to either an honest channel or one with covert messages hidden inside, and they must identify which one they see. Anamorphic cryptosystems can easily achieve this definition because they (mis)use existing cryptographic primitives which themselves have provable security guarantees — anamorphic schemes can then derive provable stealthiness from properties that lead to these very guarantees. In contrast, analysis of traditional steganography per the definition above can rely on unrealistic or difficult-to-model assumptions about a channel distribution. For instance, a steganographic channel which takes images and replaces the least significant bits of pixel colors with pseudorandom encryptions is only provably secure under the tenuous assumption that the least significant bits are uniformly distributed (and that they even exhibit a distribution that can be easily modeled in the first place).

The way anamorphic cryptosystems naturally "inherit" provable stealthiness guarantees also enables the study and construction of anamorphic schemes that achieve *extended* provable security properties against broader attacks. For example, as CCA notions of security extend CPA by considering active adversaries, we can explore stealthy channels in the presence of active censors or malicious recipients. This analysis has already been initiated with the nascent notions of robustness (discussed above) from BGHMR and private anamorphism (discussed later) from KPPYZ. It is these security notions that we critique and develop.

Anamorphic Signatures. An anamorphic signature scheme specifies (in addition to normal signature functionality) an anamorphic signing algorithm $\mathsf{asig} \leftarrow \mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$ which uses a signing key sk to sign a benign message msg and "double key" dk to hide an anamorphic message amsg inside the anamorphic signature asig. A recipient knowing dk should be able to extract amsg from asig, while one not knowing dk should not be able to distinguish asig from a real signature *even if they know* sk. This stealthiness property is formalized via a real-or-anamorphic (RoA-CAMA) game and knowledge of sk is motivated by viewing the attacker as a "dictator" who can compel disclosure of sk from their citizens.

KPPYZ construct anamorphic signature schemes for many deployed signature algorithms. We view several of their constructions as instantiations of a transform we call RRep ("randomness replacement") which utilizes signatures from which one can extract the randomness used to sign. To sign anamorphically, amsg is encrypted with dk using a pseudorandom encryption scheme prE and the resulting ciphertext is used as the randomness for a signature scheme S's signing procedure. The recipient extracts randomness from signatures and decrypts. This complete procedure forms an anamorphic signature scheme we denote $\mathsf{aS} = \mathsf{RRep}[\mathsf{S}, \mathsf{prE}]$. Since prE produces pseudorandom ciphertexts, the anamorphic signatures will be indistinguishable from honest signatures. We say that S is *randomness recovering* if it satisfies the recovery property needed for RRep. For some randomness recovering signature schemes, the signing key is needed to extract the signing randomness, while in others, the randomness is directly extractable from a signature. In the latter case, the signature scheme is denoted *publicly* randomness recovering. When RRep is applied to a plain randomness recovering signature scheme, the recipient must know sk. Such an anamorphic scheme is called "symmetric" and can only be used in a setting where the recipient is trusted to not use knowledge of sk to maliciously forge signatures. For non-symmetric schemes (e.g., RRep applied to publicly randomness recovering signature schemes), KPPYZ define a natural security goal termed private anamorphism (that should hold in addition to RoA-CAMA stealthiness), which asks that a malicious recipient, given dk and honest signatures but not sk, should not be able to forge new signatures of their own. KPPYZ prove a general result that any anamorphic signature scheme in which dk is independent of the signing key is necessarily private anamorphic (assuming the starting signature scheme was unforgeable). This captures RRep applied to publicly randomness recovering schemes as a special case.

1.1 Strengthening Robustness to Dictator Unforgeability

Robustness for Anamorphic Signatures. We begin by adapting BGHMR's robustness definition to the signature scheme setting. At a high level, robustness asks that one cannot anamorphically decrypt *honest* ciphertexts (or signatures). This property holds in addition to RoA-CAMA stealthiness (rather than instead of it). The benefits, as stated by BGHMR, are twofold. The first is practical.

> "It is intuitively desirable that a special symbol ⊥ is output indicating that the ciphertext (intentionally) contains no covert message ... [as the schemes are] potentially already being actively used for regular communication (and only occasionally required to transmit covert messages, from some point in time onward)." [BGH$^+$24, p. 5 ePrint]

The second is for security.

> "The dictator could trick receivers into reveling (*sic*) that they are indeed in possession of a double key by sending them normally encrypted messages and observing whether they show any reaction." [BGH$^+$24, p. 5 ePrint]

With robustness codified for anamorphic signature schemes, we then analyze two transforms proposed by BGHMR. The first we call RIdP (randomness identification with PRF) and the second we call RIdPX

Table 1. Existing security notions for anamorphic signatures compared to our proposed notions. Check marks (✓) denote inputs and oracle access given to adversaries. Starred check marks (✓*) denote that the adversary accesses one of two oracles in a distinguishing game. Daggered check marks ($✓^\dagger$) denote that the oracle only accepts signatures output by other oracles, as well as anamorphic decryption state.

		vk	sk	dk	O_{Sign}	O_{aSign}	O_{aDec}
Stealthiness (RoA-CAMA)	[KPP+23a]	✓	✓		✓* ≃	✓*	
Confidentiality (IND-CAMA)	[KPP+23a]	✓	✓			✓	
Confidentiality (IND-CASA)	Our Work (Sec. 3)	✓	✓			✓	✓
Robustness (ROB-CMA)	[BGH+24]				✓		$✓^\dagger$
Dictator Unforgeability (DUF-CASA)	Our Work (Sec. 4)	✓	✓		N/A	✓	✓
Private Anamorphism (PA-CMA)	[KPP+23a]	✓		✓	✓		N/A
Recipient Unforgeability (RUF-CAMA)	Our Work (Sec. 5)	✓		✓	✓	✓	N/A

(randomness identification with PRF and XOR). These transforms are both stealthy (RoA-CAMA secure) and robust. Note that, because these transforms were originally proposed for constructing anamorphic encryption, we must introduce the necessary formalism to allow them to act on signature schemes.

A Gap in Robustness. We reflect on the security motivation for robustness in the austere dictator setting envisioned for anamorphic cryptography and find that the robustness definition excludes reasonable attacks a dictator may perform. Specifically, there is no reason for the dictator to restrict themselves to sending normally encrypted messages to observe a reaction. The dictator knows the secret cryptographic key and can compute ciphertexts or signatures using whatever randomness they want (rather than using fresh randomness like in the honest case). More problematically, the dictator can take *existing* ciphertexts or signatures *which may hide encrypted anamorphic messages* and modify them before sending them to the recipient. We introduce a strengthened security definition called dictator unforgeability under chosen anamorphic signature attack to capture these scenarios. In the DUF-CASA game, the dictator is given the verification and signing keys and *separate* anamorphic signing and anamorphic decryption oracles which they can query however they want. The dictator's goal is to forge a new message-signature pair that contains an anamorphic message hidden within it. The second section of Table 1 summarizes BGHMR's robustness notion (denoted ROB-CMA) and our proposed dictator unforgeability security notion (DUF-CASA). From the table, it is evident that DUF-CASA comprehensively addresses the limitations in robustness as it gives the adversary access to all relevant values and oracles.

Breaking Transforms and Recovering Positive Results. We begin by analyzing the randomness replacement transform RRep from KPPYZ, observing that it is dictator forgeable. To address this, we propose a modification that requires the base signature scheme S to satisfy a "strong randomness recovery" property and the pseudorandom encryption scheme prE employed in the transform to be an AEAD scheme that authenticates both the anamorphic message amsg and the benign message msg. We let $\mathsf{RRep}^\star$ denote this improved transform. We then analyze the RIdP and RIdPX transforms ported from BGHMR and find that both are dictator forgeable. Fortunately, a simple modification to RIdP, which we denote $\mathsf{RIdP}^\star$, allows it to be dictator unforgeable provided the underlying signature scheme satisfies a "strong randomness identification" property. In contrast, RIdPX is unrepairable as its design principle inherently allows a dictator to easily modify an anamorphic signature to contain a different (and potentially meaningful) anamorphic message. These results are summarized in Table 2.

Extended Results. We include additional results related to dictator unforgeability that are not present in the proceedings version of this work. We introduce, in Section 3, a CCA-style notion of anamorphic message confidentiality (IND-CASA) which expands on an established (IND-CAMA) confidentiality notion discussed in KPPYZ by giving the dictator access to an anamorphic decryption oracle. We summarize both

Table 2. The security of anamorphic signature scheme transforms (adapted from BGHMR and KPPYZ) under our proposed notions. The ✓ symbol indicates security while ✗ indicates insecurity.

		RoA-CAMA	DUF-CASA	RUF-CAMA
RRep[S, prE]	[KPP$^+$23a]	✓	✗	✗
RIdP[S, prF]	[BGH$^+$24]	✓	✗	✗
RIdPX[S, prF]		✓	✗	✗
RRep*[S, prE]	Our Work	✓	✓	✓
RIdP*[S, prF]		✓	✓	✗

confidentiality notions in Table 1. We then show that anamorphic signature schemes that are IND-CAMA secure and dictator unforgeable are necessarily IND-CASA secure. In Appendix D we port our results on robustness and dictator unforgeability back to the anamorphic encryption setting.

1.2 Strengthening Private Anamorphism to Recipient Unforgeability

A Gap in Private Anamorphism. The second portion of our paper switches from the view of a dictator to the view of a malicious recipient. Our first contribution in this section shows the insufficiency of the KPPYZ notion of private anamorphism (discussed above as a property that may hold in addition to RoA-CAMA stealthiness). Recall that private anamorphism considers a recipient who is given dk and honest signatures and we desire that they cannot forge new signatures of their own. KPPYZ propose this definition so that the following holds.

> "Anamorphic messages ... come with an implicit origin authentication as they are part of a signature that can only be produced by the owner of the signing key." [KPP$^+$23a, p. 8 ePrint]

There is a crucial gap between the definition of private anamorphism and this envisioned deployment scenario. In the former, the recipient *only receives real signatures* while over the course of the latter they also see anamorphic signatures. Motivated by this, we introduce a strengthened definition called recipient unforgeability under chosen anamorphic message attack. In the RUF-CAMA game, the attacker is given the double key and oracles for requesting *both* real and anamorphic signatures. Their goal is to produce forgeries. The third section of Table 1 summarizes KPPYZ's private anamorphism (formally denoted PA-CMA) and our (RUF-CAMA) recipient unforgeability security notions. From the table we can see that RUF-CAMA patches an artificial limitation present in the definition of private anamorphism.

Breaking RRep. We start by asking whether (RUF-CAMA) recipient unforgeability is different than private anamorphism. As a simple separating example, consider an anamorphic signature scheme that appends its signing key to an anamorphic message before encrypting it. Upon seeing a single anamorphic signature, any recipient trivially recovers the sender's signing key, even though this scheme is private anamorphic. This result, however, does not resolve whether recipient unforgeability is actually *meaningfully* different from private anamorphism for *realistic* schemes. To address this, we investigate whether randomness replacement RRep from KPPYZ applied to publicly randomness-recovering schemes always results in recipient-unforgeable anamorphic signatures.

Damningly, we find this is not the case. Consider using stateful counter-mode encryption $\mathsf{prE}_{\mathsf{cm}}$ in RRep (a natural choice of scheme which provides pseudorandom ciphertexts). Then an attacker, knowing dk and the current counter value, can compute the pseudorandom pad $\mathsf{prF}(\mathsf{k},\mathsf{ctr})$ ahead of time and thus maliciously select an anamorphic message so that the resulting ciphertext $\mathsf{ct} = \mathsf{prF}(\mathsf{k},\mathsf{ctr}) \oplus \mathsf{amsg}$ equals any string of its choice. Consequently, $\mathsf{aS} = \mathsf{RRep}[\mathsf{S},\mathsf{prE}_{\mathsf{cm}}]$ using this encryption scheme is certainly insecure with any signature scheme S that is insecure against chosen randomness attacks. CRA security extends the typical unforgeability notion to allow the attacker to pick the randomness when requesting signatures. Observe that, because the attack above does not leverage any authentication property derived from dk, it extends to the RRep* transform we proposed as a dictator-unforgeable variant of RRep.

Recovering Positive Results. Next, we ask whether positive results can be recovered for the recipient unforgeability of $\mathsf{aS} = \mathsf{RRep}^\star[\mathsf{S}, \mathsf{prE}]$ with publicly randomness recovering schemes by restricting the choice of signature S or encryption scheme prE. We provide two such results by requiring a stronger security property of either the signature scheme or the encryption scheme, while assuming nothing additional of the other scheme. The first is a natural complement to our counter-mode encryption attack; if the signature scheme S is CRA secure, then $\mathsf{RRep}^\star$ is recipient unforgeable. As dk is independent of sk, all that it and anamorphic signing can do is allow the attacker to influence the randomness used by the signing algorithm. We prove CRA security of RSA-PSS (considered in KPPYZ).

This leaves unclear whether publicly randomness recovering schemes like the Boneh-Boyen signature scheme (considered in KPPYZ), which are not known to be CRA secure, can be recipient unforgeable when used in $\mathsf{RRep}^\star$. For such schemes, we take the opposite approach and assume a strong ideal-model simulatability property of the encryption scheme prE. As an example, consider an encryption scheme where $\mathsf{ct} = (r, \mathsf{msg} \oplus H(\mathsf{k}, r))$. If H is modeled as a random oracle, then using this in $\mathsf{RRep}^\star$ with any publicly randomness-recovering unforgeable signature scheme S (which doesn't itself use H) will be recipient unforgeable. The forging reduction adversary simulates H internally and can emulate anamorphic signatures using normal signatures by extracting ct from the normal signature and reprogramming H to be consistent. We make this ideal-model analysis modular by extracting this core reprogrammability property to define a new notion of simulatability with random ciphertexts ($\mathsf{SIM\text{-}\$CT}$) which will be achieved by appropriate idealizations of typical randomized encryption schemes like counter-mode encryption or cipher block chaining. Our positive results for $\mathsf{RRep}^\star$ are included in Table 2.

Dictator and Recipient Unforgeable Schemes. Users are likely to require anamorphic signature schemes that provide *both* dictator unforgeability and recipient unforgeability. Summarizing the results above, our improved $\mathsf{RRep}^\star$ transform satisfies both properties. Turning to RIdP^*, which we modified to achieve dictator unforgeability, we demonstrate, through a concrete attack, that it fails to achieve recipient unforgeability when applied to signature schemes of interest. Moreover, the same attack applies to RIdPX, revealing that this previously proposed scheme permits anamorphic signatures that are forgeable by both dictators *and* recipients. This occurs despite RIdPX being designed to achieve robustness and satisfying private anamorphism. These results are summarized in Table 2.

1.3 Related Work

Conceptually, anamorphic cryptography (broadly construed) is deeply related to a variety of research areas such as steganography and subliminal channels [Cra98,Sim83,YY04] and kleptography/algorithmic substitution attacks [AMV20,BPR14,BJK15,YY97]. These connections can be useful for inspiring positive results (e.g., RRep is reminiscent of the IV-replacement attack of [BPR14]). In the other direction, negative results could also likely be ported over. For example, its possible that anamorphic signatures could be somewhat prevented if users are required to use signature schemes with unique signatures [BPR14,DFP15] or for which cryptographic reverse firewalls [MS15] are known. While techniques transfer between these areas, the change in high level perspective also inspires different low level technical insights (e.g., CRA-secure signatures and $\mathsf{SIM\text{-}\$CT}$-secure encryption).

1.4 Subsequent Work

This is the full version of [JS24] which we will refer to as the proceedings version. Differences between the versions are summarized after the references. After the publication of the proceedings version, two works have proposed definitions analogous to our dictator unforgeability notion.

In [DG25], the authors formalize three worlds which capture dictators with varying levels of control over the cryptography their citizens use. Their main technical result is anamorphic-resistant encryption, for which practical-bandwidth anamorphic channels are uninstantiatable under strong dictators who influence public parameters. On the other hand, they observe the same insufficiencies of robustness as we do and propose *anamorphic unforgeability*, equivalent to our dictator unforgeability notion for anamorphic encryption.[1] They show that a variant of $\mathsf{RRep}^\star$ (which uses random seeds and a MAC, while our construction uses an AEAD scheme) and anamorphic Naor-Yung satisfy this notion.

In [CCLZ25], the authors explore a litany of security definitions for anamorphic encryption and establish implications and separations between these definitions and provide one construction that achieves the strongest security they consider. This construction is essentially $\mathsf{RRep}^\star$ that injects additional randomness. They also show that dictator unforgeability, which they call $\mathsf{ROB\text{-}CT+}$, along with stealthiness, implies a form of CCA stealthiness. This is analogous to our result that dictator unforgeability and $\mathsf{IND\text{-}CAMA}$ security implies $\mathsf{IND\text{-}CASA}/\mathsf{IND\text{-}CACA}$ security. They also generalize dictator unforgeability to asymmetric anamorphic encryption, proposed by [CGM24].

2 Notation and Preliminaries

2.1 Pseudocode, Sets, and Tables

We leverage pseudocode formalism [BR06]. For an algorithm A, we write $y \leftarrow A^O(x)$ to denote running A with oracle access to O on input x and assigning the output to y. When A is stateful, $A(x : \mathsf{st})$ denotes running A on input x and state st, where st is passed by reference and can be updated by A. When A is probabilistic, $A(x; r)$ denotes running A on input x and seed r. Omitting r indicates that the seed is sampled uniformly. We write PPT for probabilistic polynomial time.

Adversaries (e.g., $\mathcal{A}$, $\mathcal{B}$) are (tuples of) algorithms which may make up to $q(\lambda)$ oracle queries given a security parameter λ. We will write $q(\lambda)$ as q. Due to identically named oracles in some proofs, we notationally clarify them by writing $O^{\mathcal{A}}_{\mathsf{Alg}}$ to denote the Alg oracle that an adversary $\mathcal{A}$ accesses. Given a security notion SEC, we may consider modified notions where the adversary can access oracles for underlying primitives (e.g., in the random oracle model). We notate this as $\mathsf{SEC}(O_1, O_2, \dots)$. For a game $\mathcal{G}$, we let $\Pr[\mathcal{G}(\lambda) \Rightarrow 1]$ denote the probability $\mathcal{G}$ outputs 1 for a security parameter λ.

For a scheme C, parameters pp (which we discuss shortly), and associated class of objects X (e.g., keys, messages, or signatures), we call the set of possible values the X space (key space, message space, signature space) and denote these with double struck capital letters ($\mathsf{C}.\mathbb{K}_{\mathsf{pp}}$, $\mathsf{C}.\mathbb{M}_{\mathsf{pp}}$, $\mathsf{C}.\mathbb{S}_{\mathsf{pp}}$). We implicitly assume algorithms check whether their inputs are in the appropriate spaces, rejecting if not, and correspondingly assume that attackers do not provide inputs to oracles that are not in the appropriate space. For a stateful algorithm $\mathsf{C}.\mathsf{Alg}(\dots : \mathsf{st})$, we denote the set of possible states st with $\mathsf{C}.\mathbb{ST}^{\mathsf{Alg}}_{\mathsf{pp}}$. For a probabilistic algorithm $\mathsf{C}.\mathsf{Alg}(\dots; r)$, we denote the set of possible random seeds r with $\mathsf{C}.\mathbb{R}^{\mathsf{Alg}}_{\mathsf{pp}}$. We abuse notation and write $|\mathsf{C}.\mathbb{X}_{\mathsf{pp}}|$ to be both the size of $\mathsf{C}.\mathbb{X}_{\mathsf{pp}}$ for $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$ and a bound thereon that is a function of λ. Depending on the context, it should be a polynomial upper bound or a superpolynomial bound.

For a distribution χ over a set $\mathbb{S}$, we write $s \leftarrow\!\!\$\ \chi(\mathbb{S})$ to denote that s is sampled from χ or $s \leftarrow\!\!\$\ \mathbb{S}$ if χ is uniform over $\mathbb{S}$. For a table T, we write $\mathsf{T}[x]$ to denote the element indexed by x or $\perp$ if no value has yet been assigned. We assume this indexing process is PPT. Tables are automatically initialized upon first assignment. For a list L, we let $\mathsf{L}.\mathsf{add}(x)$ be a PPT algorithm that appends x to L. We denote the i^{th} element of L with $\mathsf{L}[i]$ (i.e., lists are one-indexed) and the last element with $\mathsf{L}[-1]$. The empty list is denoted $[\cdot]$. For any set, table, or list X, we let $|X|$ denote the number of elements. We say that a function $f : \mathbb{A} \to \mathbb{B} \cup \{\perp\}$ is a quasi-injection if $f(x) = f(y) \neq \perp$ implies $x = y$ for all $x, y \in \mathbb{A}$, i.e., if f is an injection on all inputs that are not mapped to $\perp$.

We make use of the Fundamental Lemma of Game Playing from [BR06]. If games $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$ are identical except after a boolean flag bad is set to true, then

$$\Pr[\mathcal{G}_i(\lambda) \Rightarrow 1] - \Pr[\mathcal{G}_{i+1}(\lambda) \Rightarrow 1] \leq \Pr[\mathcal{G}_i(\lambda) \text{ sets } \mathsf{bad}] = \Pr[\mathcal{G}_{i+1}(\lambda) \text{ sets } \mathsf{bad}].$$

2.2 Cryptographic Primitives and Preliminaries

We proceed with the understanding that the reader is familiar with digital signature schemes, pseudorandom functions, and pseudorandom encryption. For digital signature schemes we consider strong

[1] As we discuss in in Section 4.5, this definition differs only syntactically (but not from a security perspective) between anamorphic signatures and encryption because the dictator receives both the public and secret keys.

unforgeability under chosen message attack ($\mathsf{SUF\text{-}CMA}$). For pseudorandom functions we consider pseudorandom function security (PRF). Pseudorandom encryption refers to a randomized or stateful symmetric encryption scheme that has ciphertexts indistinguishable from random under chosen plaintext attack ($\mathsf{IND\$\text{-}CPA}$) and which may achieve ciphertext integrity ($\mathsf{INT\text{-}CT}$). Such a scheme may take in associated data, which it can ignore or authenticate (e.g., the pseudorandom encryption scheme is an AEAD scheme). Full details are provided in Appendix A.

Several of the schemes we discuss require public parameters. Due to interdependence between the parameter generation of schemes used together, we simplify notation by defining a global parameter generation algorithm $\mathsf{PublicParamGen}(1^\lambda)$ that takes in a security parameter λ and outputs parameters pp (implicitly) usable by all of the primitives and constructions in this work. One can think of pp capturing specifications like concrete groups and lengths of hash function inputs and outputs that are baked into global standards outside of any individual's control. We let $\mathbb{PP}$ denote the set of all possible parameters.

Stateful encryption and triviality predicates. We will define CCA- and unforgeability-style security notions for stateful schemes. In these definitions, an adversary does not win if it trivially forwards ciphertexts or signatures it has received without modification. Stateful schemes vary on what "unmodified" means. To illustrate, let $\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \ldots$ be a sequence of ciphertexts output by a CCA-secure encryption scheme SE. For certain SE, reordering the ciphertexts (e.g. $\mathsf{ct}_3, \mathsf{ct}_1, \mathsf{ct}_2, \ldots$) *does not* count as modification so a reordered sequence is considered trivial and returns $\perp$ when queried to the decryption oracle of a CCA game. For other SE, reordering *does* count as modification so the game returns the decryption of a reordered sequence when queried to a decryption oracle as it is not considered trivial. Similarly, various SE may differ on whether replay attacks (e.g. $\mathsf{ct}_1, \mathsf{ct}_1, \mathsf{ct}_2, \ldots$) count as modification or not, which affects when the game returns $\perp$. As these are only two examples out of many possibilities, we parameterize games with a triviality predicate $\mathsf{pred}_{\mathsf{trivial}}$ that checks, given a list $\mathsf{L}_{\mathsf{Enc}}$ of ciphertexts output by encryption and a list $\mathsf{L}_{\mathsf{Dec}}$ of ciphertexts input to decryption (both with their associated data), whether the adversary $\mathcal{A}$ makes a trivial query to a relevant decryption oracle or outputs a trivial attempted forgery. When $\mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}})$ outputs 1, the oracle returns $\perp$ or the game does not count the attempt as a successful forgery, respectively.

In discussion, it will be useful to distinguish between *strict* schemes, which permanently reject all inputs following any rejected input (e.g. a ciphertext with invalid authentication tag) and *permissive* schemes, which process subsequent inputs as if the rejected inputs were never observed. A given scheme must follow one of these conventions (which affects the $\mathsf{pred}_{\mathsf{trivial}}$-based security we should expect from it), but it is often possible to generically switch between the two. When describing concrete instantiations, we will default to permissive schemes because they are the most compatible with BGHMR's [BGH+24] envisioned scenario where a user is unsure whether a given ciphertext contains an anamorphic message.

The most permissive predicate we will consider is $\mathsf{pred}^{\mathsf{perm}}_{\mathsf{trivial}}$, which returns 1 if the last element of $\mathsf{L}_{\mathsf{Dec}}$ appears somewhere in $\mathsf{L}_{\mathsf{Enc}}$. This predicate allows arbitrary reordering and replay of ciphertexts. The most strict predicate we will consider is $\mathsf{pred}^{\mathsf{sync}}_{\mathsf{trivial}}$, which returns 1 if $\mathsf{L}_{\mathsf{Dec}}$ is a prefix of $\mathsf{L}_{\mathsf{Enc}}$. This predicate does not allow any reordering or replay of ciphertexts. In general, we expect that $[\![\mathsf{pred}^{\mathsf{sync}}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}}) = 1]\!] \implies [\![\mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}}) = 1]\!] \implies [\![\mathsf{pred}^{\mathsf{perm}}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}}) = 1]\!]$ will hold for any $\mathsf{pred}_{\mathsf{trivial}}$ worth considering. See the discussion of encryption in Appendix A.4 for further details and discussion on triviality predicates and strict/permissive schemes.

2.3 Randomness Recovery and Identification for Signature Schemes

We will analyze multiple constructions arrived at by applying generic transformations to signature schemes which allow for recovering or identifying the randomness used for signing. We define both properties of signature schemes below and show that certain schemes considered by Kutylowski, Persiano, Phan, Yung, and Zawada (KPPYZ) [KPP+23a] exhibit these properties. Note that KPPYZ implicitly consider randomness recovery (and define it for "three-message public-coin protocols"), though they do not use our modular formalism. They did not consider randomness identification.

Table 3. Randomness recovering and identifying signature schemes

Signature Scheme	Recovering	Publicly Recovering	Identifying
ElGamal [ElG84]	✓	✗	✓
Schnorr [Sch91]	✓	✗	✓
RSA-PSS [BR96]	✓	✓	✓
Boneh-Boyen [BB04]	✓	✓	✓

Definition 1 (Randomness Recovery). *A signature scheme* S *is* randomness recovering *if it additionally specifies a PPT algorithm* $\mathsf{S.RRecov}$ *such that, for all* $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$, $\mathsf{msg} \in \mathsf{S}.\mathbb{M}_{\mathsf{pp}}$, $r \in \mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}$, *and* $\mathsf{sig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$, r *can be recovered by computing* $r \leftarrow \mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{sig})$.

For some schemes, $\mathsf{S.RRecov}$ can efficiently recover the randomness without the signing key sk. In this case, we say S is *publicly randomness recovering* and omit sk as an input to the recovery algorithm. We provide examples of randomness recovering signature schemes in Table 3, which also notes if they are publicly randomness recovering.

In certain proofs, our use of the randomness recovery property does not require the function $\mathsf{S.RRecov}$ to be PPT. In such cases, we say S is *(in)efficiently randomness recovering.* As we will shortly discuss, an (in)efficient $\mathsf{S.RRecov}$ can often be constructed generically.

Definition 2 (Randomness Identification). *A signature scheme* S *is* randomness identifying *if it additionally specifies a PPT algorithm* $\mathsf{S.RIdtfy}$ *such that, for all* $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$, $\mathsf{msg} \in \mathsf{S}.\mathbb{M}_{\mathsf{pp}}$, $r, r' \in \mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}$, *and* $\mathsf{sig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$, $\mathsf{S.RIdtfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}, r')$ *outputs* 1 *if and only if* $r' = r$.

We do not consider schemes that require sk to identify the randomness. Given $\mathsf{S.RIdtfy}$, one can construct an inefficient $\mathsf{S.RRecov}$ that iterates over all possible randomness $r' \in \mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}$ until it finds one such that $\mathsf{S.RIdtfy}$ outputs 1. Thereby all randomness identifying signature schemes can also be viewed as (in)efficiently randomness recovering schemes. As shown in Table 3, all signature schemes we consider are randomness identifying.

Definitions 1 and 2 suffice for the correctness and basic security of the constructions we analyze; however, a new security definition we will introduce requires the stronger property that randomness recovery ensures only a single signature sig can result in a given randomness r being recovered (for fixed vk, sk, and msg). For our ultimate use of this, it will be useful to state this property in terms of quasi-injections (functions that are injections on all inputs not mapped to $\perp$).

Definition 3 (Strong Randomness Recovery). *A signature scheme* S *is* strongly randomness recovering *if it is randomness recovering and for all* $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$, *and* $\mathsf{msg} \in \mathsf{S}.\mathbb{M}_{\mathsf{pp}}$, *the function* $f : \widehat{\mathsf{sig}} \mapsto \mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \widehat{\mathsf{sig}})$ *is a quasi-injection.*

Strong randomness recovery doesn't follow from the plain randomness recovery as the latter only considers honestly generated signatures $\mathsf{sig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$. The strong definition also considers signatures that might never be output by the signing procedure. For signature schemes where the signing key sk is used to recover randomness, we can generically achieve strong randomness recovery by having $\mathsf{S.RRecov}$ run $\mathsf{sig}' \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$ and return $\perp$ if $\mathsf{sig}' \neq \mathsf{sig}$. For publicly recovering signature schemes, there is no such a generic fix. However, for the specific schemes we consider, it suffices to have $\mathsf{S.RRecov}$ check if the signature is valid by running $\mathsf{S.Vrfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$ and returning $\perp$ if it is invalid. This technique relies on the fact that algorithms are implicitly assumed to check that their inputs are from the correct spaces. Below, we define a strong randomness identification property (which does not need sk) for schemes that are strongly randomness recovering (which may need sk).

Definition 4 (Strong Randomness Identification). *A signature scheme* S *is* strongly randomness identifying *if it is randomness identifying and (in)efficiently strongly randomness recovering, specifying*

$\mathsf{S.RIdtfy}$ *and* $\mathsf{S.RRecov}$ *such that, for all* $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$*,* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$*,* $\mathsf{msg} \in \mathsf{S}.\mathbb{M}_{\mathsf{pp}}$*,* $\mathsf{sig} \in \mathsf{S}.\mathbb{S}_{\mathsf{pp}}$*, and* $r \in \mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}$*,* $\mathsf{S.RIdtfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}, r)$ *outputs 1 if and only if* $r' = r$ *where* $r' \leftarrow \mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{sig})$ *and* $r' \neq \perp$*.*

We now discuss concrete (plain) signature schemes which will serve as useful examples throughout this work. For two schemes, the ElGamal signature scheme and RSA-PSS, we provide full descriptions and additionally specify their randomness recovery and identification algorithms.

ElGamal Signature Scheme. For a security parameter λ, suppose that the public parameter generation algorithm $\mathsf{PublicParamGen}(1^\lambda)$ outputs pp containing a prime p where $\log(p) \in \Theta(\lambda)$, a generator g of $\mathbb{Z}_p^*$, and a hash function $H : \{0,1\}^* \times \mathbb{Z}_p^* \to \mathbb{Z}_p$. Then the ElGamal signature scheme ElG [ElG84,PS96] is as defined in Figure 1.

In the same figure, we show the algorithms $\mathsf{ElG.RRecov}$ and $\mathsf{ElG.RIdtfy}$ which make ElG randomness recovering and randomness identifying respectively. By using the technique discussed above, ElG is *strongly* randomness recovering. Furthermore, it is *strongly* randomness identifying. In particular, observe that if $\mathsf{ElG.RIdtfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}, r') = 1$ where $\mathsf{vk} = g^x$ and $\mathsf{sig} = (R, s)$, then $g^{r'} = R$ and $g^{\mathsf{ch}} = y^R R^s \pmod p$ for $\mathsf{ch} = H(\mathsf{msg}, R)$ since verification succeeds. It follows that $g^{\mathsf{ch}} = g^{xR} \cdot g^{r's} \pmod p$ so $\mathsf{ch} = xR + r's \pmod{p-1}$ since g is a generator of $\mathbb{Z}_p^*$, hence $s = (\mathsf{ch} - xR)r'^{-1}$. As $r' \in \mathbb{Z}_{p-1}^*$ (per the implicit input space check), then $\mathsf{ElG.Sign}(\mathsf{sk}, \mathsf{msg}; r') = \mathsf{sig}$ and hence $\mathsf{ElG.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{sig}) = r'$. Conversely, given $\mathsf{ElG.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{sig}) = r' \neq \perp$ it is clear that $\mathsf{sig} = (R, s)$ verifies (by correctness, since it can be generated by $\mathsf{S.Sign}$) and similarly $g^{r'} = R$, thus $\mathsf{ElG.RIdtfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}, r') = 1$.

RSA-PSS. Let $\mathbb{P}_\ell$ denote the set of ℓ-bit primes and $\mathsf{RSA.eGen}$ output RSA public exponents. For a security parameter λ, suppose $\mathsf{PublicParamGen}(1^\lambda)$ outputs pp containing positive integers λ_0, λ_1 such that $\lambda_0 + \lambda_1 \leq \lambda - 1$ and hash functions $H : \{0,1\}^* \times \{0,1\}^{\lambda_0} \to \{0,1\}^{\lambda_1}$ and $G : \{0,1\}^{\lambda_1} \to \{0,1\}^{\lambda - \lambda_1 - 1}$. We consider G as separate G_1 and G_2 which output the first λ_0 and remaining $\lambda - \lambda_0 - \lambda_1 - 1$ bits of G's outputs respectively. Then RSA-PSS [BR96] is as defined in Figure 1.

In the same figure, we show the algorithm $\mathsf{RSA\text{-}PSS.RRecov}$ which makes it publicly randomness recovering and the trivial $\mathsf{RSA\text{-}PSS.RIdtfy}$ derived therefrom which makes it randomness identifying. Observe that $\mathsf{RSA\text{-}PSS}$ is *strongly* randomness recovering as $\mathsf{RSA\text{-}PSS.RRecov}$, for fixed and honestly generated vk and fixed msg, is a quasi-injection. To establish this, we need only show that it is injective over *valid* signatures because it internally runs $\mathsf{RSA\text{-}PSS.Vrfy}$ and returns $\perp$ if verification fails. Let $r = r'$ be two randomnesses recovered using $\mathsf{RSA\text{-}PSS.RRecov}$ on valid signatures sig and sig'. It follows that $w = H(\mathsf{msg}, r) = H(\mathsf{msg}, r') = w'$, $\alpha = G(w) \oplus r = G_1(w') \oplus r' = \alpha'$ and $\gamma = G_2(w) = G_2(w') = \gamma'$. Hence, $\mathsf{sig} = (0 \| w \| \alpha \| \gamma)^e \pmod N = (0 \| w' \| \alpha' \| \gamma')^e \pmod N = \mathsf{sig}'$.

Other Schemes. KPPYZ highlight several additional signature schemes that are randomness recovering, including Boneh-Boyen [BB08], RSA-PFDH [Cor02], DSA [Nat23a], Probabilistic-Rabin [BR96], and Schnorr [Sch91] (and more generally, any signature scheme obtained by applying Fiat-Shamir to randomness recovering "three-message public-coin protocols"). We will discuss the Boneh-Boyen signature scheme in Section 5.6, where we provide relevant background.

3 Anamorphic Signature Schemes

3.1 Syntax and Security of Anamorphic Signatures

We now review anamorphic signature schemes, introduced by Kutylowski, Persiano, Phan, Yung, and Zawada (KPPYZ) [KPP+23a]. At a high level, the objective of anamorphic signatures is to appear like regular signatures except that a user knowing a secret symmetric key, called a double key dk, can encrypt a covert message into a signature asig. This covert message, called the anamorphic message amsg, can be recovered from the asig using dk.

```
$\mathsf{ElG}.\mathbb{VK}_{\mathsf{pp}} = \mathbb{PP} \times \mathbb{Z}_p^*$
$\mathsf{ElG}.\mathbb{SK}_{\mathsf{pp}} = \mathbb{PP} \times (\mathbb{Z}_{p-2} \setminus \{0\})$
$\mathsf{ElG}.\mathbb{M}_{\mathsf{pp}} = \{0,1\}^*$
$\mathsf{ElG}.\mathbb{S}_{\mathsf{pp}} = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$
$\mathsf{ElG}.\mathbb{R}_{\mathsf{pp}}^{\mathsf{Sign}} = \mathbb{Z}_{p-1}^*$

$\mathsf{ElG}.\mathsf{KeyGen}(\mathsf{pp})$
$x \leftarrow\!\!\$\ \mathbb{Z}_{p-2} \setminus \{0\}$
$y \leftarrow g^x \pmod p$
$\mathsf{vk} \leftarrow (\mathsf{pp}, y)$
$\mathsf{sk} \leftarrow (\mathsf{pp}, x)$
$\mathbf{return}\ (\mathsf{vk}, \mathsf{sk})$

$\mathsf{ElG}.\mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$
$(\mathsf{pp}, x) \leftarrow \mathsf{sk}$
$r \leftarrow\!\!\$\ \mathbb{Z}_{p-1}^*$
$R \leftarrow g^r \pmod p$
$\mathsf{ch} \leftarrow H(\mathsf{msg}, R)$
$s \leftarrow (\mathsf{ch} - xR)r^{-1} \pmod{p-1}$
$\mathsf{sig} \leftarrow (R, s)$
$\mathbf{return}\ \mathsf{sig}$

$\mathsf{ElG}.\mathsf{Vrfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$
$(\mathsf{pp}, y) \leftarrow \mathsf{vk}$
$(R, s) \leftarrow \mathsf{sig}$
$\mathsf{ch} \leftarrow H(\mathsf{msg}, R)$
$\mathbf{return}\ [\![g^{\mathsf{ch}} = y^R R^s \pmod p]\!]$

$\mathsf{ElG}.\mathsf{RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{sig})$
$(\mathsf{pp}, x) \leftarrow \mathsf{sk}$
$(R, s) \leftarrow \mathsf{sig}$
$r \leftarrow (H(\mathsf{msg}, R) - xR)s^{-1}$
$\mathbf{if}\ \mathsf{ElG}.\mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r) \neq \mathsf{sig}\ \mathbf{then}$
    $\mathbf{return}\ \perp$
$\mathbf{return}\ r$

$\mathsf{ElG}.\mathsf{RIdtfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}, r')$
$\mathbf{if}\ \mathsf{ElG}.\mathsf{Vrfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) \neq 1\ \mathbf{then}$
    $\mathbf{return}\ 0$
$(R, s) \leftarrow \mathsf{sig}$
$\mathbf{return}\ [\![g^{r'} = R \pmod p]\!]$
```

```
$\mathsf{RSA\text{-}PSS}.\mathbb{VK}_{\mathsf{pp}} = \mathbb{PP} \times \{0,1\}^\lambda \times \{0,1\}^\lambda$
$\mathsf{RSA\text{-}PSS}.\mathbb{SK}_{\mathsf{pp}} = \mathbb{PP} \times \{0,1\}^\lambda \times \{0,1\}^\lambda$
$\mathsf{RSA\text{-}PSS}.\mathbb{M}_{\mathsf{pp}} = \{0,1\}^*$
$\mathsf{RSA\text{-}PSS}.\mathbb{S}_{\mathsf{pp}} = \{0,1\}^\lambda$
$\mathsf{RSA\text{-}PSS}.\mathbb{R}_{\mathsf{pp}}^{\mathsf{Sign}} = \{0,1\}^{\lambda_0}$

$\mathsf{RSA\text{-}PSS}.\mathsf{KeyGen}(\mathsf{pp})$
$p \neq q \leftarrow\!\!\$\ \mathbb{P}_{\lambda/2}$
$N \leftarrow pq$
$e \leftarrow \mathsf{RSA}.\mathsf{eGen}(\mathsf{pp}, \varphi(N))$
$d \leftarrow e^{-1} \pmod{\varphi(N)}$
$\mathsf{vk} \leftarrow (\mathsf{pp}, N, e)$
$\mathsf{sk} \leftarrow (\mathsf{pp}, N, d)$
$\mathbf{return}\ (\mathsf{vk}, \mathsf{sk})$

$\mathsf{RSA\text{-}PSS}.\mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$
$(\mathsf{pp}, N, d) \leftarrow \mathsf{sk}$
$r \leftarrow\!\!\$\ \{0,1\}^{\lambda_0}$
$w \leftarrow H(\mathsf{msg}, r)$
$\alpha \leftarrow G_1(w) \oplus r$
$\gamma \leftarrow G_2(w)$
$\mathsf{sig} \leftarrow (0\|w\|\alpha\|\gamma)^d \pmod N$
$\mathbf{return}\ \mathsf{sig}$

$\mathsf{RSA\text{-}PSS}.\mathsf{Vrfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$
$(\mathsf{pp}, N, e) \leftarrow \mathsf{vk}$
$b\|w\|\alpha\|\gamma \leftarrow \mathsf{sig}^e \pmod N$
$r \leftarrow G_1(w) \oplus \alpha$
$\mathbf{return}\ [\![b = 0]\!] \wedge [\![w = H(\mathsf{msg}, r)]\!] \wedge [\![\gamma = G_2(w)]\!]$

$\mathsf{RSA\text{-}PSS}.\mathsf{RRecov}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$
$\mathbf{if}\ \mathsf{RSA\text{-}PSS}.\mathsf{Vrfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) \neq 1\ \mathbf{then}$
    $\mathbf{return}\ \perp$
$(\mathsf{pp}, N, e) \leftarrow \mathsf{vk}$
$b\|w\|\alpha\|\gamma \leftarrow \mathsf{sig}^e \pmod N$
$r \leftarrow G_1(w) \oplus \alpha$
$\mathbf{return}\ r$

$\mathsf{RSA\text{-}PSS}.\mathsf{RIdtfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}, r')$
$r \leftarrow \mathsf{RSA\text{-}PSS}.\mathsf{RRecov}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$
$\mathbf{return}\ [\![r' = r]\!]$
```

Fig. 1. ElGamal signature scheme (left) and RSA-PSS (right)

These anamorphic messages should be undetectable by a "dictator" who can see all signatures, force parties to hand over their secret signing keys, and ask parties to sign messages of the dictator's choosing. To realize this, the signature scheme must still sign and verify like a standard signature scheme. Furthermore, the verification key, signing key, and signatures must be indistinguishable from honestly generated ones. Below, we define the syntax for stateful anamorphic signature schemes. We will omit the states as inputs to or outputs of algorithms when considering stateless anamorphic signature schemes.

Definition 5 (Anamorphic Signature Scheme). *An* anamorphic signature scheme aS *is a signature scheme (specifying* $\mathsf{aS.KeyGen}$*,* $\mathsf{aS.Sign}$*, and* $\mathsf{aS.Vrfy}$*) with three additional PPT algorithms.*

- $\mathsf{aS.aKeyGen}(\mathsf{pp})$ *takes public parameters* pp *and generates a signing key* $\mathsf{sk} \in \mathsf{aS}.\mathbb{SK}_{\mathsf{pp}}$*, verification key* $\mathsf{vk} \in \mathsf{aS}.\mathbb{VK}_{\mathsf{pp}}$*, double key* $\mathsf{dk} \in \mathsf{aS}.\mathbb{DK}_{\mathsf{pp}}$*, initial anamorphic signing state* $\mathsf{st}_{\mathsf{aSign}} \in \mathsf{aS}.\mathbb{ST}^{\mathsf{aSign}}_{\mathsf{pp}}$*, and initial anamorphic decryption state* $\mathsf{st}_{\mathsf{aDec}} \in \mathsf{aS}.\mathbb{ST}^{\mathsf{aDec}}_{\mathsf{pp}}$*. It outputs* $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aSign}}, \mathsf{st}_{\mathsf{aDec}})$*.*
- $\mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aSign}})$ *takes a signing key* $\mathsf{sk} \in \mathsf{aS}.\mathbb{SK}_{\mathsf{pp}}$*, double key* $\mathsf{dk} \in \mathsf{aS}.\mathbb{DK}_{\mathsf{pp}}$*, message* $\mathsf{msg} \in \mathsf{aS}.\mathbb{M}_{\mathsf{pp}}$*, anamorphic message* $\mathsf{amsg} \in \mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}$*, and anamorphic signing state* $\mathsf{st}_{\mathsf{aSign}} \in \mathsf{aS}.\mathbb{ST}^{\mathsf{aSign}}_{\mathsf{pp}}$ *and outputs an anamorphic signature* $\mathsf{asig} \in \mathsf{aS}.\mathbb{AS}_{\mathsf{pp}}$*. It also updates the state* $\mathsf{st}_{\mathsf{aSign}}$*.*
- $\mathsf{aS.aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig} : \mathsf{st}_{\mathsf{aDec}})$ *takes a verification key* $\mathsf{vk} \in \mathsf{aS}.\mathbb{VK}_{\mathsf{pp}}$*, double key* $\mathsf{dk} \in \mathsf{aS}.\mathbb{DK}_{\mathsf{pp}}$*, message* $\mathsf{msg} \in \mathsf{aS}.\mathbb{M}_{\mathsf{pp}}$*, anamorphic signature* $\mathsf{sig} \in \mathsf{aS}.\mathbb{AS}_{\mathsf{pp}}$*, and anamorphic decryption state* $\mathsf{st}_{\mathsf{aDec}} \in \mathsf{aS}.\mathbb{ST}^{\mathsf{aDec}}_{\mathsf{pp}}$ *and outputs an anamorphic message* $\mathsf{amsg} \in \mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}$*. It also updates the state* $\mathsf{st}_{\mathsf{aDec}}$*.*

In certain anamorphic signature schemes, dk includes sk. KPPYZ refer to such schemes as *symmetric*, distinguishing them from all other schemes, which we label *non-symmetric*. In addition to standard signature scheme correctness, anamorphic schemes must satisfy the additional requirement that anamorphic messages are properly decryptable from anamorphic signatures. We adopt a game-based approach to define correctness as we deal with stateful schemes. This is formalized in Figure 2 as the $\mathcal{G}^{\mathsf{CORR}}$ game. Essentially, we require that anamorphic messages signed in a specific order should anamorphically decrypt to the correct anamorphic messages when decrypted in the same order. This correctness property is rather narrow, and various applications or schemes may rely on broader notions of correctness (as we discuss for symmetric encryption in Appendix A.4). The CORR advantage for an anamorphic signature scheme aS is defined as

$$\mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{aS},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{CORR}}_{\mathsf{aS},\mathcal{A}}(\lambda) \Rightarrow 1].$$

Definition 6 (Correctness). *An anamorphic signature scheme* aS *is* correct *if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{aS},\mathcal{A}}(\lambda)$ *is negligible.*

In Section 2 we mentioned *strict* schemes, which permanently reject all inputs after encountering any invalid input, and *permissive* schemes, which process subsequent inputs as though the rejected inputs had not occurred. Below, we formally define the two conventions for anamorphic signatures.

Definition 7 (Strictness/Permissiveness). *An anamorphic signature scheme* aS *is* permissive *if* $\mathsf{st}_{\mathsf{aDec}}$ *is unmodified when computing* $\mathsf{aS.aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig} : \mathsf{st}_{\mathsf{aDec}})$ *and the output* $\mathsf{amsg} = \bot$*. An anamorphic signature scheme* aS *is* strict *if* $\mathsf{st}_{\mathsf{aDec}}$ *is set to* $\bot$ *in the same situation and* $\mathsf{aS.aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig} : \bot)$ *always outputs* $\bot$*.*

We now examine measures of bandwidth for anamorphic signature schemes and discuss how these considerations are reflected in our chosen syntax, before moving on to security.

Anamorphic Length Efficiency. Anamorphic channels (whether over signatures, public key encryption, or other mechanisms) are often restricted in how much anamorphic data can be inserted per signature/ciphertext. Thus, it is valuable to design anamorphic schemes which are as efficient as possible in how much data they can fit to improve bandwidth. To simplify discussion, we will talk of this efficiency asymptotically. We say that the size of an anamorphic message space $\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}$ is polynomial or exponential (in λ) corresponding to the "length" of anamorphic messages being logarithmic or polynomial (typically linear) respectively. This asymptotic perspective is useful for broad understanding of how

```
$\mathcal{G}^{\mathsf{CORR}}_{\mathsf{aS},\mathcal{A}}(\lambda)$
    $\mathsf{win} \leftarrow 0$
    $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
    $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aSign}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow \mathsf{aS.KeyGen}(\mathsf{pp})$
    $\textbf{run } \mathcal{A}^{O_{\mathsf{aSign},\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
    $\textbf{return } \mathsf{win}$
```

```
$O_{\mathsf{aSign},\mathsf{aDec}}(\mathsf{msg}, \mathsf{amsg})$
    $\mathsf{asig} \leftarrow \mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aSign}})$
    $\mathsf{amsg}^* \leftarrow \mathsf{aS.aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig} : \mathsf{st}_{\mathsf{aDec}})$
    $\mathsf{win} \leftarrow \mathsf{win} \vee [\![\mathsf{amsg} \neq \mathsf{amsg}^*]\!]$
    $\textbf{return } \mathsf{asig}$
```

```
$\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$
    $b \leftarrow\!\!\$\ \{0,1\}$
    $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
    $\textbf{if } b = 0 \textbf{ then}$
        $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{aS.KeyGen}(\mathsf{pp})$
    $\textbf{else}$
        $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aSign}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow \mathsf{aS.aKeyGen}(\mathsf{pp})$
    $b^* \leftarrow \mathcal{A}^{O_{\mathsf{Sign}/\mathsf{aSign}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
    $\textbf{return } [\![b = b^*]\!]$
```

```
$O_{\mathsf{Sign}/\mathsf{aSign}}(\mathsf{msg}, \mathsf{amsg})$
    $\textbf{if } b = 0 \textbf{ then}$
        $\mathsf{sig} \leftarrow \mathsf{aS.Sign}(\mathsf{sk}, \mathsf{msg})$
        $\textbf{return } \mathsf{sig}$
    $\textbf{else}$
        $\mathsf{asig} \leftarrow \mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aSign}})$
        $\textbf{return } \mathsf{asig}$
```

```
$\mathcal{G}^{\mathsf{IND\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1)}(\lambda)$ $\mathcal{G}^{\mathsf{IND\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1)}(\lambda)$
    $b \leftarrow\!\!\$\ \{0,1\}$
    $(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}}) \leftarrow ([\cdot], [\cdot])$
    $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
    $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aSign}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow \mathsf{aS.aKeyGen}(\mathsf{pp})$
    $(\mathsf{msg}^*, \mathsf{amsg}^*_0, \mathsf{amsg}^*_1, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_0^{O_{\mathsf{aSign}}, O_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
    $\mathsf{asig}^* \leftarrow \mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}^*, \mathsf{amsg}^*_b : \mathsf{st}_{\mathsf{aSign}})$
    $\mathsf{L}_{\mathsf{aSign}}.\mathsf{add}((\mathsf{msg}^*, \mathsf{asig}^*))$
    $b^* \leftarrow \mathcal{A}_1^{O_{\mathsf{aSign}}, O_{\mathsf{aDec}}}(\mathsf{st}_{\mathcal{A}}, \mathsf{asig}^*)$
    $\textbf{return } [\![b = b^*]\!]$
```

```
$O_{\mathsf{aSign}}(\mathsf{msg}, \mathsf{amsg})$
    $\mathsf{asig} \leftarrow \mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aSign}})$
    $\mathsf{L}_{\mathsf{aSign}}.\mathsf{add}((\mathsf{msg}, \mathsf{asig}))$
    $\textbf{return } \mathsf{asig}$

$O_{\mathsf{aDec}}(\mathsf{msg}, \mathsf{asig})$
    $\mathsf{amsg} \leftarrow \mathsf{aS.aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig} : \mathsf{st}_{\mathsf{aDec}})$
    $\mathsf{L}_{\mathsf{aDec}}.\mathsf{add}((\mathsf{msg}, \mathsf{asig}))$
    $\textbf{if } \mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}}) = 1 \textbf{ then}$
        $\textbf{return } \bot$
    $\textbf{return } \mathsf{amsg}$
```

Fig. 2. Anamorphic signature scheme correctness game $\mathcal{G}^{\mathsf{CORR}}$ (top), stealthiness game $\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}$ (middle), and confidentiality games $\mathcal{G}^{\mathsf{IND\text{-}CAMA}}$ and $\mathcal{G}^{\mathsf{IND\text{-}CASA}}$ (bottom)

generic anamorphic techniques compare with each other (e.g. polynomial message space will always be worse than exponential). However, it is an imperfect proxy for reality where "linear" length messages would be a few hundred bits for a signature scheme based on elliptic curve groups, but a few thousand bits for signature schemes based on finite fields, RSA, or lattices.

Some prior works on anamorphic cryptography measure this efficiency in terms of ratios between anamorphic message "length" and some parameter of the original scheme (e.g. the length of the non-anamorphic message [PPY22] or the difference in length between the ciphertext and the non-anamorphic message [BGH$^+$24]). For similar reasons, these measurements are nice proxies when comparing different anamorphic transforms applied to a fixed underlying non-anamorphic scheme; however, they are quite loose when comparing anamorphic techniques across different underlying schemes.

All constructions we consider can broadly be viewed as some sort of insertion of pseudorandom ciphertexts (hiding the anamorphic message) into the randomness of the signature schemes. We will thereby lose a polynomial number of bits of anamorphic message "length" if these pseudorandom ciphertexts are randomized or need to include authentication tags. The former can sometimes be avoided by using a stateful, deterministic scheme (which is one motivation behind our stateful formalism). Unfortunately, the latter is unavoidable for some of our results given our "atomic" formalism where one anamorphic message must be stored in one anamorphic signature. If anamorphic cryptography were deployed in practice, the deployer should consider a "non-atomic" approach where authentication occurs over time. We elect not to leverage non-atomic formalism for anamorphic messages as we focus on the core motivation behind our security notions and because such approaches require labor-intensive formalism that should be chosen based on low-level details of how the anamorphic signatures will be deployed. See [FGMP15] for a general discussion on this topic.

Anamorphic Stealthiness. We now shift from bandwidth to security, beginning with stealthiness. As mentioned prior, a dictator should not be able to tell whether plain signing or anamorphic signing is employed. This is formalized by KPPYZ via the real-or-anamorphic game $\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}$ defined in Figure 2, which asks that a dictator with knowledge of a party's verification key, signing key, and signatures on messages and anamorphic messages of their choosing cannot tell if they receive values output by honest key generation and signing or instead anamorphic key generation and anamorphic signing. The $\mathsf{RoA\text{-}CAMA}$ advantage for an anamorphic signature scheme aS is defined as

$$\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) = 2\Pr[\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \Rightarrow 1] - 1.$$

Definition 8 (Stealthiness). *An anamorphic signature scheme* aS *is stealthy (in the* $\mathsf{RoA\text{-}CAMA}$ *sense) if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$ *is negligible.*

Anamorphic Message Confidentiality. In addition to stealthiness, parties desire that anamorphic messages encrypted within anamorphic signatures are confidential, which KPPYZ capture with a CPA-style anamorphic message indistinguishability under chosen anamorphic message attack game $\mathcal{G}^{\mathsf{IND\text{-}CAMA}}$ shown in Figure 2 (without highlighted portions). Here, the adversary receives verification and signing keys but not the double key, selects a benign message and two anamorphic messages, and must guess (with access to an anamorphic signing oracle) which anamorphic message is encrypted in a signature they receive. The $\mathsf{IND\text{-}CAMA}$ advantage for an anamorphic signature scheme aS is defined as

$$\mathbf{Adv}^{\mathsf{IND\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) = 2\Pr[\mathcal{G}^{\mathsf{IND\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \Rightarrow 1] - 1.$$

Definition 9 (IND-CAMA Confidentiality). *An anamorphic signature scheme* aS *is confidential (in the* $\mathsf{IND\text{-}CAMA}$ *sense) if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{IND\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$ *is negligible.*

KPPYZ prove that ($\mathsf{RoA\text{-}CAMA}$) stealthy anamorphic signature schemes are ($\mathsf{IND\text{-}CAMA}$) confidential, so it is sufficient to analyze only $\mathsf{RoA\text{-}CAMA}$ security. At a high level, this result follows from the fact that a PPT adversary $\mathcal{A}$ against the $\mathsf{IND\text{-}CAMA}$ security of an anamorphic signature scheme aS can be used to construct a PPT $\mathcal{B}$ against the $\mathsf{RoA\text{-}CAMA}$ security of aS that randomly samples a bit b', queries its signing oracle on input $(\mathsf{msg}, \mathsf{amsg}_{b'})$, and receives asig which it sends to $\mathcal{A}$. When $\mathcal{B}$ receives real signatures $\mathcal{A}$'s view is completely independent of b', while when $\mathcal{B}$ receives anamorphic signatures it simulates the $\mathcal{G}^{\mathsf{IND\text{-}CAMA}}$ game to $\mathcal{A}$.

Theorem 1 (Theorem 1 of [KPP+23a]). *Let* aS *be a stealthy anamorphic signature scheme. Then it is also* $\mathsf{IND\text{-}CAMA}$ *confidential. In particular, for all PPT adversaries* $\mathcal{A}$*, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq 2\,\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aS},\mathcal{B}}(\lambda).$$

While $\mathsf{IND\text{-}CAMA}$ captures passive dictators who simply view anamorphic signatures, an *active* dictator may try to break anamorphic message confidentiality by taking previously sent anamorphic signatures, modifying them, sending them to parties, and observing how those parties respond. We thus introduce a natural CCA-style notion captured by the anamorphic message indistinguishability under chosen anamorphic signature attack game $\mathcal{G}^{\mathsf{IND\text{-}CASA}}$ shown in Figure 2 (with highlighted portions). Here,

$\mathsf{aS}.\mathbb{DK}_{\mathsf{pp}} = \mathsf{prE}.\mathbb{K}_{\mathsf{pp}} \boxed{\times \mathsf{S}.\mathbb{SK}_{\mathsf{pp}}}$
$\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}} = \mathsf{prE}.\mathbb{M}_{\mathsf{pp}}$
$\mathsf{aS}.\mathbb{ST}^{\mathsf{aSign}}_{\mathsf{pp}} = \mathsf{prE}.\mathbb{ST}^{\mathsf{Enc}}_{\mathsf{pp}}$
$\mathsf{aS}.\mathbb{ST}^{\mathsf{aDec}}_{\mathsf{pp}} = \mathsf{prE}.\mathbb{ST}^{\mathsf{Dec}}_{\mathsf{pp}}$

```
$\mathsf{aS.aKeyGen}(\mathsf{pp})$
    $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
    $(\mathsf{k}, \mathsf{st}_{\mathsf{Enc}}, \mathsf{st}_{\mathsf{Dec}}) \leftarrow \mathsf{prE.KeyGen}(\mathsf{pp})$
    $\mathsf{dk} \leftarrow (\mathsf{k}, \boxed{\mathsf{sk}})$
    $(\mathsf{st}_{\mathsf{aSign}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow (\mathsf{st}_{\mathsf{Enc}}, \mathsf{st}_{\mathsf{Dec}})$
    return $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aSign}}, \mathsf{st}_{\mathsf{aDec}})$
```

$\mathsf{aS.KeyGen} = \mathsf{S.KeyGen}$
$\mathsf{aS.Sign} = \mathsf{S.Sign}$
$\mathsf{aS.Vrfy} = \mathsf{S.Vrfy}$

```
$\mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk} = (\mathsf{k}, \boxed{\mathsf{sk}}), \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aSign}})$
    $\mathsf{act} \leftarrow \mathsf{prE.Enc}(\mathsf{k}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aSign}})$
    $\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; \mathsf{act})$
    return $\mathsf{asig}$

$\mathsf{aS.aDec}(\mathsf{vk}, \mathsf{dk} = (\mathsf{k}, \boxed{\mathsf{sk}}), \mathsf{msg}, \mathsf{asig} : \mathsf{st}_{\mathsf{aDec}})$
    $\mathsf{act} \leftarrow \mathsf{S.RRecov}(\mathsf{vk}, \boxed{\mathsf{sk}}, \mathsf{msg}, \mathsf{asig})$
    $\mathsf{amsg} \leftarrow \mathsf{prE.Dec}(\mathsf{k}, \mathsf{msg}, \mathsf{act} : \mathsf{st}_{\mathsf{aDec}})$
    return $\mathsf{amsg}$
```

Fig. 3. Randomness replacement transforms RRep [KPP+23a] and RRep*

the adversary receives verification and signing keys, selects a benign message and two anamorphic messages, and must guess (with access to an anamorphic signing *and* decryption oracles) which anamorphic message is encrypted in a signature they receive. Note that the anamorphic decryption oracle contains a triviality check using a triviality predicate as discussed in Section 2.2. The IND-CASA advantage for an anamorphic signature scheme aS and triviality predicate $\mathsf{pred}_{\mathsf{trivial}}$ is defined as

$$\mathbf{Adv}^{\mathsf{IND\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) = 2\Pr[\mathcal{G}^{\mathsf{IND\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \Rightarrow 1] - 1.$$

Definition 10 (IND-CASA Confidentiality). *An anamorphic signature scheme* aS *is confidential (in the* IND-CASA *sense) for a triviality predicate* $\mathsf{pred}_{\mathsf{trivial}}$ *if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{IND\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda)$ *is negligible.*

Analogous to KPPYZ's result that stealthiness implies CPA-style anamorphic message confidentiality (IND-CAMA), we could consider the relationship between some form of stealthiness under active dictators and the CCA-style anamorphic message confidentiality (IND-CASA) introduced above. While there are several ways to define a security game for stealthiness under active dictators, they all must specify how the anamorphic decryption oracle operates in the "real" world where anamorphic signatures are not employed. Philosophically, it is unclear what it means for a party to anamorphically decrypt when they are not using anamorphic signatures and don't have a double key (and likely don't even know about the existence of anamorphic cryptography).

3.2 Constructing Anamorphic Signatures

Anamorphism via Randomness Replacement. KPPYZ introduce a diverse collection of ways to construct anamorphic signatures, including the Fiat-Shamir transform, rejection sampling, and the Naor-Yung paradigm. We view many of their proposed signature schemes as instantiations of a randomness replacement transform which can construct anamorphic signature schemes from a myriad of well-studied and widely-deployed signature schemes.

Randomness replacement exploits the probabilistic nature of many signature schemes by encoding covert messages in the random coins used to generate signatures. If the signature schemes are randomness recovering (Definition 1), then these random coins can be recovered by a recipient with sufficient knowledge and hence enable extraction of the covert message. Concretely, anamorphic signature schemes constructed via randomness replacement encrypt anamorphic messages into pseudorandom ciphertexts which are then used as randomness when generating a signature. This randomness can then be recovered by a recipient through the randomness recovery function and decrypted to obtain the anamorphic message. KPPYZ's anamorphic signature schemes based on the Boneh-Boyen, ElGamal, DSA, RSA-PSS and Schnorr can all be viewed as instances of this technique. The formal construction capturing KPPYZ's schemes is as given below. We also specify our proposed $\mathsf{RRep}^\star$ (first discussed in the Introduction) in the same figure, though we defer discussion of the modification to Section 4.

Construction 1 (RRep and $\mathsf{RRep}^\star$). *Consider the following primitives and requirements needed to construct an anamorphic signature scheme via randomness replacement.*

- *Let* S *be a strongly randomness-recovering signature scheme with recovery function* $\mathsf{S.RRecov}$.
- *Let* prE *be a pseudorandom encryption scheme supporting associated data.*
- *Let* $\mathsf{PublicParamGen}$ *output parameters* pp *such that* $\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}} = \mathsf{prE}.\mathbb{C}_{\mathsf{pp}}$ *and* $\mathsf{S}.\mathbb{M}_{\mathsf{pp}} = \mathsf{prE}.\mathbb{AD}_{\mathsf{pp}}$.

Then the anamorphic signature schemes $\mathsf{aS} = \mathsf{RRep}[\mathsf{S}, \mathsf{prE}]$ *(no highlighted parts) and* $\mathsf{aS} = \mathsf{RRep}^\star[\mathsf{S}, \mathsf{prE}]$ *(with highlighted parts) are constructed as shown in Figure 3, where in both cases the boxed code may be omitted if* S *is publicly randomness recovering.*

Construction 1 requires that the given encryption scheme is pseudorandom with respect to the signature scheme's randomness space. We assume suitable encryption schemes exist, as they can likely be achieved for most signature schemes using existing encoding and rejection sampling techniques. In decryption, act might equal $\perp$ (for example, if sig is an invalid signature). The understanding is that $\mathsf{prE.Dec}$ outputs $\mathsf{amsg} = \perp$ in this case, additionally setting $\mathsf{st}_{\mathsf{aDec}} \leftarrow \perp$ if it is strict or leaving $\mathsf{st}_{\mathsf{aDec}}$ unmodified if it is permissive. Overall, randomness replacement transforms result in stealthy anamorphic signature schemes (Definition 8). Since we formalize the construction slightly differently from KPPYZ, we accordingly provide adapted versions of their Theorem 9 and 10 (which cover stealthiness) in Appendix B.1.

When RRep and $\mathsf{RRep}^\star$ are applied to signature schemes such as ElGamal (Figure 1) where RRecov requires sk, the resulting anamorphic signature schemes are symmetric. When the underlying schemes are publicly randomness recovering, the resulting anamorphic signature schemes are non-symmetric. For example, RRep or $\mathsf{RRep}^\star$ applied to RSA-PSS (Figure 1) is non-symmetric.

Anamorphism via Rejection Sampling. We briefly discuss rejection sampling, touched on in KPPYZ, which generically constructs a non-symmetric anamorphic signature scheme aS from *any* (sufficiently) probabilistic signature scheme (e.g., ElGamal). The idea is to sample signatures sig until $\mathsf{prF}(\mathsf{k}, \mathsf{sig}) = \mathsf{amsg}$ for some pseudorandom function prF. This signature is returned as the anamorphic signature asig and anamorphic decryption simply evaluates $\mathsf{amsg} \leftarrow \mathsf{prF}(\mathsf{k}, \mathsf{asig})$. Observe that the size of the anamorphic message space $\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}$ must be adequately small for anamorphic signing to run in polynomial time. While we don't formally analyze rejection sampling in this work, it will serve as a useful comparison for other anamorphic constructions.

Anamorphism via Randomness Identification with PRF. We adapt and discuss in detail two (of many) anamorphic transforms proposed by Banfi, Gegier, Hirt, Maurer, and Rito (BGHMR) [BGH+24] to the signature scheme setting. In contrast to rejection sampling, in which the sender "searches" for a signature that encrypts the anamorphic message they want to send, these transforms transfer this workload to the recipient who, given a signature, "searches" for the right anamorphic message encrypted within it. BGHMR's transforms were designed to achieve a robustness security notion; however, we defer detailed discussion of this to Section 4 and focus here on presenting the transforms.

We begin by reviewing BGHMR's first construction, originally described for anamorphic encryption, before porting it to anamorphic signature schemes. Their construction transforms any $\mathsf{IND\text{-}CPA}$-secure public-key encryption scheme PKE to a stateful anamorphic encryption scheme aPKE and roughly follows a decrypt/re-encrypt paradigm. The double key dk consists solely of a pseudorandom function key k. Users also maintain synchronized counters $\mathsf{ctr}_{\mathsf{aEnc}}$ and $\mathsf{ctr}_{\mathsf{aDec}}$. To perform anamorphic encryption, the sender computes $r \leftarrow \mathsf{prF}(\mathsf{k}, (\mathsf{ctr}_{\mathsf{aEnc}}, \mathsf{amsg}))$ and returns $\mathsf{act} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}; r)$. This procedure also updates the encryption counter $\mathsf{ctr}_{\mathsf{aEnc}} \leftarrow \mathsf{ctr}_{\mathsf{aEnc}} + 1$. To perform anamorphic decryption of act, the receiver

```
$\mathsf{aS}.\mathbb{DK}_{\mathsf{pp}} = \mathsf{aS}.\mathbb{VK}_{\mathsf{pp}} \times \mathsf{prF}.\mathbb{K}_{\mathsf{pp}}$
$\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}} = \mathsf{prF}.\mathbb{R}_{\mathsf{pp}}$
$\mathsf{aS}.\mathbb{ST}^{\mathsf{aSign}}_{\mathsf{pp}} = \mathsf{aS}.\mathbb{ST}^{\mathsf{aDec}}_{\mathsf{pp}} = \mathbb{Z}_n$

$\mathsf{aS.KeyGen} = \mathsf{S.KeyGen}$
$\mathsf{aS.Sign} = \mathsf{S.Sign}$
$\mathsf{aS.Vrfy} = \mathsf{S.Vrfy}$

$\mathsf{aS.aKeyGen}(\mathsf{pp})$
  $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
  $\mathsf{dk} \leftarrow \mathsf{prF.KeyGen}(\mathsf{pp})$
  $\mathsf{ctr}_{\mathsf{aSign}} \leftarrow 1$
  $\mathsf{ctr}_{\mathsf{aDec}} \leftarrow 1$
  return $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}, \mathsf{ctr}_{\mathsf{aSign}}, \mathsf{ctr}_{\mathsf{aDec}})$

$\mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{ctr}_{\mathsf{aSign}})$
  $r \leftarrow \boxed{\mathsf{prF}(\mathsf{dk}, (\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{msg}, \mathsf{amsg}))}$
       $\mathsf{amsg} \oplus \mathsf{prF}(\mathsf{dk}, \mathsf{ctr}_{\mathsf{aSign}})$
  $\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$
  $\mathsf{ctr}_{\mathsf{aSign}} \leftarrow \mathsf{ctr}_{\mathsf{aSign}} + 1$
  return $\mathsf{asig}$

$\mathsf{aS.aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig} : \mathsf{ctr}_{\mathsf{aDec}})$
  for $\mathsf{amsg} \in \mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}$ do
    $r' \leftarrow \boxed{\mathsf{prF}(\mathsf{dk}, (\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{msg}, \mathsf{amsg}))}$
         $\mathsf{amsg} \oplus \mathsf{prF}(\mathsf{dk}, \mathsf{ctr}_{\mathsf{aDec}})$
    if $\mathsf{S.RIdtfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{asig}, r') = 1$ then
      $\mathsf{ctr}_{\mathsf{aDec}} \leftarrow \mathsf{ctr}_{\mathsf{aDec}} + 1$
      return $\mathsf{amsg}$
  return $\perp$
```

Fig. 4. Randomness identification transforms $\boxed{\mathsf{RIdP}}$, $\boxed{\mathsf{RIdP}^\star}$ [BGH+24], and RIdPX

decrypts using sk to obtain msg and re-encrypts msg using randomness $r' \leftarrow \mathsf{prF}(\mathsf{k}, (\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{amsg}))$ for all $\mathsf{amsg} \in \mathsf{aPKE}.\mathbb{AM}_{\mathsf{pp}}$. Once the receiver finds an amsg that encrypts to act, it returns this amsg, after which it increments $\mathsf{ctr}_{\mathsf{aDec}}$, or it returns $\perp$ if no such amsg exists. Note that, similar to rejection sampling, the anamorphic message space $\mathsf{aPKE}.\mathbb{AM}_{\mathsf{pp}}$ must be sufficiently small for the scheme to run in polynomial time. Furthermore, both sender and receiver must ensure that their counters $\mathsf{ctr}_{\mathsf{aEnc}}$ and $\mathsf{ctr}_{\mathsf{aDec}}$ are synchronized for correctness to hold.

In the anamorphic signature scheme setting, the recipient of an anamorphic signature may not have access to the sender's signing key (i.e., non-symmetric anamorphism) and hence cannot directly perform a re-signing procedure analogous to re-encryption. However, the transform above still functions even if the recipient only *identifies* whether a given seed was used as randomness during encryption (re-encryption is simply a way to realize this). Thus, the idea behind the transform can be applied to any signature scheme that is randomness identifying (Definition 2) without knowledge of the signing key.

We can now describe the adapted first transform from BGHMR, which we denote as randomness identification with PRF (RIdP). We also adapt BGHMR's variant of the RIdP transform that doesn't input the anamorphic message into the pseudorandom function but instead XORs it (which they do for efficiency). We denote this as randomness identification with PRF and XOR (RIdPX). Finally, we also specify our proposed $\mathsf{RIdP}^\star$ transform (first discussed in the Introduction) in the same construction, though we again defer discussion of the modification to Section 4.

Construction 2 (RIdP, $\mathsf{RIdP}^\star$, and RIdPX). *Consider the following primitives and requirements needed to construct an anamorphic signature scheme via randomness identification with PRF (and XOR).*

- *Let* S *be a strongly randomness-identifying signature scheme with identification function* $\mathsf{S.RIdtfy}$.
- *Let* prF *be a pseudorandom function that takes in* $\boxed{\textit{2-tuple}}$, $\boxed{\textit{3-tuple}}$, *or* *1-tuple* *messages where the first element of the tuple is an integer.*
- *Let* $\mathsf{PublicParamGen}$ *output parameters* pp *such that* $\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}} = \mathsf{prF}.\mathbb{R}_{\mathsf{pp}}$ *where these sets form a group over* $\oplus$ *and* $\mathsf{prF}.\mathbb{M}_{\mathsf{pp}} = \boxed{\mathbb{Z}_n \times \mathsf{S}.\mathbb{M}_{\mathsf{pp}} \times \mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}}$ *where* n *is a positive integer defined in* pp.

```
$\mathsf{prE.KeyGen}(\mathsf{pp})$
  $\mathsf{k} \leftarrow \mathsf{prF.KeyGen}(\mathsf{pp})$
  $(\mathsf{ctr}_{\mathsf{Enc}}, \mathsf{ctr}_{\mathsf{Dec}}) \leftarrow (1, 1)$
  $\mathbf{return}\ (\mathsf{k}, \mathsf{ctr}_{\mathsf{Enc}}, \mathsf{ctr}_{\mathsf{Dec}})$

$\mathsf{prE.Enc}(\mathsf{k}, \mathsf{ad}, \mathsf{msg} : \mathsf{ctr}_{\mathsf{Enc}})$
  $\mathsf{ct} \leftarrow \boxed{\mathsf{prF}(\mathsf{k}, (\mathsf{ctr}_{\mathsf{Enc}}, \colorbox{lightgray}{$\mathsf{ad}$}, \mathsf{msg}))}$
         $\dbox{$\mathsf{msg} \oplus \mathsf{prF}(\mathsf{k}, \mathsf{ctr}_{\mathsf{Enc}})$}$
  $\mathsf{ctr}_{\mathsf{Enc}} \leftarrow \mathsf{ctr}_{\mathsf{Enc}} + 1$
  $\mathbf{return}\ \mathsf{ct}$

$\mathsf{prE.Dec}(\mathsf{k}, \mathsf{ad}, \mathsf{ct} : \mathsf{ctr}_{\mathsf{Dec}})$
  $\mathbf{for}\ \mathsf{msg} \in \mathsf{prE}.\mathbb{M}_{\mathsf{pp}}\ \mathbf{do}$
    $\mathsf{ct}' \leftarrow \boxed{\mathsf{prF}(\mathsf{k}, (\mathsf{ctr}_{\mathsf{Dec}}, \colorbox{lightgray}{$\mathsf{ad}$}, \mathsf{msg}))}$
           $\dbox{$\mathsf{msg} \oplus \mathsf{prF}(\mathsf{k}, \mathsf{ctr}_{\mathsf{aDec}})$}$
    $\mathbf{if}\ \mathsf{ct} = \mathsf{ct}'\ \mathbf{then}$
      $\mathsf{ctr}_{\mathsf{Dec}} \leftarrow \mathsf{ctr}_{\mathsf{Dec}} + 1$
      $\mathbf{return}\ \mathsf{msg}$
  $\mathbf{return}\ \bot$
```

Fig. 5. Pseudorandom encryption schemes for instantiating $\boxed{\mathsf{RIdP}}$, $\boxed{\colorbox{lightgray}{$\mathsf{RIdP}^\star$}}$, and $\dbox{$\mathsf{RIdPX}$}$ from RRep or $\colorbox{lightgray}{$\mathsf{RRep}^\star$}$

Then the anamorphic signature schemes $\boxed{\mathsf{aS} = \mathsf{RIdP}[\mathsf{S}, \mathsf{prF}]}$ *and* $\dbox{$\mathsf{aS} = \mathsf{RIdPX}[\mathsf{S}, \mathsf{prF}]$}$ *(no highlighted parts) and* $\boxed{\mathsf{aS} = \mathsf{RIdP}^\star[\mathsf{S}, \mathsf{prF}]}$ *(with highlighted parts) are constructed as shown in Figure 4.*

By using the inefficient randomness recovery algorithm induced by applying the randomness identification algorithm $\mathsf{S.RIdtfy}$ on all possible randomness, we can view RIdP, RIdPX, and $\mathsf{RIdP}^\star$ as instances of RRep or $\mathsf{RRep}^\star$ (with inefficient decryption) using the pseudorandom encryption schemes shown in Figure 5. Thereby, many of our coming results for RIdP, RIdPX, or $\mathsf{RIdP}^\star$ can be captured as special cases of our results for RRep or $\mathsf{RRep}^\star$ by proving appropriate properties of the encryption schemes. For concreteness, we will instead directly write proofs for these anamorphic schemes (which implicitly embed the analysis of the underlying encryption schemes).

The ported RIdP and RIdPX transforms achieve stealthiness (Definition 8) and the robustness property introduced by BGHMR which we discuss shortly. We provide full proofs of both in Appendix B.2.

4 Strengthening Robustness to Dictator Unforgeability

Following the introduction of anamorphic encryption [PPY22], Banfi, Gegier, Hirt, Maurer, and Rito (BGHMR) [BGH+24] proposed a security notion dubbed robustness which allows parties communicating via anamorphic encryption to identify ciphertexts containing anamorphic messages. As discussed in the Introduction, BGHMR present both practical and security considerations as motivation for robustness. A network with anamorphic channels presumably contains both honest and anamorphic ciphertexts, so an anamorphic party must be able to systematically distinguish such ciphertexts. Furthermore, a dictator may attempt to discern whether a party is using an anamorphic channel by sending them fresh ciphertexts, and hence a party should only decrypt anamorphic messages that were truly sent.

Because robustness was proposed for anamorphic encryption schemes, we first adapt and formalize robustness to the anamorphic signature scheme setting before proposing a stronger notion.

4.1 Robustness (for Anamorphic Signatures)

The anamorphic signature scheme variant of robustness is captured by the $\mathcal{G}^{\mathsf{ROB\text{-}CMA}}$ game shown in Figure 6. Following BGHMR, we present it as distinguishing game and consequently define the $\mathsf{ROB\text{-}CMA}$ advantage for an anamorphic signature scheme aS by

$$\mathbf{Adv}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) = 2\Pr[\mathcal{G}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \Rightarrow 1] - 1.$$

Definition 11 (Robustness). *An anamorphic signature scheme* aS *is robust (in the* $\mathsf{ROB\text{-}CMA}$ *sense) if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$ *is negligible.*

```
$\mathcal{G}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$
  $b \leftarrow\!\!\$ \{0,1\}$
  $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^{\lambda})$
  $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aSign}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow \mathsf{aS.aKeyGen}(\mathsf{pp})$
  $b^* \leftarrow \mathcal{A}^{O_{\mathsf{Sign},\mathsf{aDec}}}(\mathsf{pp})$
  return $[\![b = b^*]\!]$
```

```
$O_{\mathsf{Sign},\mathsf{aDec}}(\mathsf{msg}, \mathsf{st}^*_{\mathsf{aDec}})$
  if $b = 0$ then
    return $\bot$
  $\mathsf{sig} \leftarrow \mathsf{aS.Sign}(\mathsf{sk}, \mathsf{msg})$
  $\mathsf{amsg} \leftarrow \mathsf{aS.aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{sig} : \mathsf{st}^*_{\mathsf{aDec}})$
  return $\mathsf{amsg}$
```

Fig. 6. Robustness game $\mathcal{G}^{\mathsf{ROB\text{-}CMA}}$ adapted from [BGH$^+$24]

The BGHMR robustness notion considers an adversary not privy to *any* keys that views attempted anamorphic decryptions of *fresh, honest* ciphertexts. The $\mathcal{G}^{\mathsf{ROB\text{-}CMA}}$ game in Figure 6 captures this by giving the adversary access to an equivalent combined oracle $O_{\mathsf{Sign},\mathsf{aDec}}$ that signs new signatures and attempts to anamorphically decrypt them.[2] An adversary easily wins if any decryption is successful as it can trivially see that $b = 1$.

In some ways, this definition is artificially weak. The attacker does not even know vk and thus cannot pick messages that depend on it. In some ways, it is artificially strong. The attacker can choose to set $\mathsf{st}^*_{\mathsf{aDec}}$ arbitrarily, say, to states that would never occur while actually running anamorphic decryption. These observations alone are not major flaws of the robustness definition because, for many constructions, robustness relies on non-cryptographic counting arguments and hence modifications to the specifics of the definition would not alter the conclusions for these schemes. However, as we will shortly discuss, robustness is insufficient in other respects.

Before continuing, we first recall that BGHMR proposed robustness alongside a litany of transforms including (the anamorphic encryption versions of) RIdP and RIdPX. They proved robustness for these schemes because (the anamorphic encryption versions of) RRep and rejections sampling *don't* achieve robustness. While we present a modified $\mathsf{RRep}^{\star}$ transform that can be instantiated to use an authenticated pseudorandom encryption scheme, the nature of rejection sampling appears to prohibit it from achieving robustness. This is because providing a signature as input to a pseudorandom function always results in some sort of output anamorphic message, and the inherent restriction on anamorphic message space size makes message authentication difficult.

4.2 Dictator Unforgeability

We revisit the rationale behind robustness as discussed in BGHMR, specifically keeping in mind the dictator setting within which anamorphic cryptography aims to be secure. To reiterate, the first motivation is practical — many honest and anamorphic cryptographic outputs will be transmitted in a given network so parties should be able to systematically identify when a given ciphertext (or signature in our case) contains an anamorphic message encrypted within it.[3] Robustness captures this goal. The second motivation is for security — a party should only be able to perform anamorphic decryption on ciphertexts (or signatures) which were truly anamorphically encrypted by another party, as a dictator might send ciphertexts or signatures to parties and see how they react. Robustness falls short of capturing this by excluding many practical actions a dictator may take.

In the public-key encryption setting considered in BGHMR, a dictator knows the public key and can encrypt messages themselves using randomness of their own choosing before sending them to users. The robustness game does not give the public key to the dictator and only computes fresh encryptions, hence precluding any attacks where a dictator selects particular randomness. This form of attack also applies

[2] In Table 1, we presented this oracle as separate O_{Sign} and O_{aDec} oracles where the latter only takes signatures output by the former. The formulation in Figure 6 is equivalent.

[3] This motivation makes sense for permissive schemes (Definition 7) used in settings where the environment does not afford an external means of identifying which ciphertexts/signatures have anamorphic content.

```
$\mathcal{G}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda)$
    $(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}}) \leftarrow ([\cdot], [\cdot])$
    $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^{\lambda})$
    $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aSign}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow \mathsf{aS.aKeyGen}(\mathsf{pp})$
    $(\mathsf{msg}^*, \mathsf{asig}^*) \leftarrow \mathcal{A}^{O_{\mathsf{aSign}}, O_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
    $\mathsf{L}_{\mathsf{aDec}}.\mathsf{add}((\mathsf{msg}^*, \mathsf{asig}^*))$
    $b_{\mathsf{valid}} \leftarrow [\![\mathsf{aS.aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}^*, \mathsf{asig}^* : \mathsf{st}_{\mathsf{aDec}}) \neq \bot]\!]$
    $b_{\mathsf{trivial}} \leftarrow \mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}})$
    return $b_{\mathsf{valid}} \wedge \neg b_{\mathsf{trivial}}$
```

```
$O_{\mathsf{aSign}}(\mathsf{msg}, \mathsf{amsg})$
    $\mathsf{asig} \leftarrow \mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aSign}})$
    $\mathsf{L}_{\mathsf{aSign}}.\mathsf{add}((\mathsf{msg}, \mathsf{asig}))$
    return $\mathsf{asig}$

$O_{\mathsf{aDec}}(\mathsf{msg}, \mathsf{asig})$
    $\mathsf{amsg} \leftarrow \mathsf{aS.aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig} : \mathsf{st}_{\mathsf{aDec}})$
    $\mathsf{L}_{\mathsf{aDec}}.\mathsf{add}((\mathsf{msg}, \mathsf{asig}))$
    return $\mathsf{amsg}$
```

Fig. 7. Dictator unforgeability game $\mathcal{G}^{\mathsf{DUF\text{-}CASA}}$

to signature schemes as the anamorphic setting considers a dictator who obtains secret keys (again not given in BGHMR's definition) and can thus sign messages with randomness of their choosing.

Even more critically, it is unreasonable for a dictator to restrict themselves solely to generating new ciphertexts or signatures. Instead, the dictator may take existing ciphertexts or signatures, which may contain anamorphic messages that they have influenced, and maul them into other ciphertexts or signatures the dictator then uses for an attack. For instance, suppose the dictator's actions have caused a user to covertly send the message "Let's meet at 2:00PM" via an anamorphic signature. The dictator may modify the signature in transit so that the anamorphic message now says "Let's meet at 4:00PM" without even needing conclusive knowledge that an anamorphic message is hidden inside.

To address these gaps between the robustness notion and desired security in the anamorphic setting, we introduce a new ciphertext-integrity-inspired security notion which we call dictator unforgeability under chosen anamorphic signature attack (DUF-CASA). In this notion, a dictator gets access to an *anamorphic* signing oracle and may request anamorphic decryptions on any (real or anamorphic) signatures it obtains or generates itself with the goal of producing a *new* signature that doesn't anamorphically decrypt to $\bot$. This is formalized in the $\mathcal{G}^{\mathsf{DUF\text{-}CASA}}$ game in Figure 7. As the dictator now has access to an anamorphic signing oracle, they may try to return an unmodified anamorphic signature as their forgery. Since we deal with stateful schemes, we detect this using a triviality predicate $\mathsf{pred}_{\mathsf{trivial}}$ as discussed in Section 2. Unlike robustness, which is defined as an indistinguishability notion, we capture our security goal as an unforgeability game. Hence, we define the DUF-CASA advantage for an anamorphic signature scheme aS and triviality predicate $\mathsf{pred}_{\mathsf{trivial}}$ by

$$\mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \Rightarrow 1].$$

Definition 12 (Dictator Unforgeability). *An anamorphic signature scheme* aS *is dictator unforgeable (in the* DUF-CASA *sense) for a triviality predicate* $\mathsf{pred}_{\mathsf{trivial}}$ *if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda)$ *is negligible.*

Considering this definition with $\mathsf{pred}^{\mathsf{perm}}_{\mathsf{trivial}}$ (which returns 1 if the last element of $\mathsf{L}_{\mathsf{Dec}}$ appears somewhere in $\mathsf{L}_{\mathsf{Enc}}$) gives the definition of dictator unforgeability in our proceedings version [JS24].

By including msg in $\mathsf{L}_{\mathsf{aSign}}$ and $\mathsf{L}_{\mathsf{aDec}}$, we require the anamorphic operations to authenticate the non-anamorphic message. This seems prudent in case the user of the anamorphic cryptography will somehow take this message into account when interpreting the meaning of the anamorphic message. If we defined security without including msg in these lists, then the types of schemes we consider — that simply replace signature randomness with some sort encryption of the anamorphic message — would anyway need to authenticate msg to prevent "re-signing" attacks that create a new signature that decrypts by signing a different message with the randomness from an anamorphic signature. None of our definitions consider using the same dk with multiple different signing keys (a possibility mentioned by KPPYZ [KPP+23a]), but in such a setting one would likely want to authenticate vk as well.

Dictator unforgeability is not strictly stronger than robustness due to an artificial reason involving statefulness. In particular, while it strengthens robustness by additionally giving the adversary both verification and signing keys, an anamorphic signing oracle, and anamorphic decryptions on any anamorphic signature of its choosing, it does not allow the adversary to select the anamorphic decryption state as in robustness. Directly selecting the decryption state (possibly to a state that would never arise in honest use) does not correspond to a capability we expect attackers to have in practice.

Rather unsurprisingly, anamorphic signature schemes that are dictator unforgeable (which we mentioned earlier is inspired by ciphertext integrity) and achieve CPA-style chosen anamorphic message confidentiality (IND-CAMA) necessarily achieve CCA-style chosen anamorphic *signature* confidentiality (IND-CASA). This is formalized in the following theorem and a full proof is given in Appendix C.1.

Theorem 2 (IND-CAMA + DUF-CASA ⇒ IND-CASA). *Let* aS *be an* IND-CAMA *indistinguishable and dictator unforgeable anamorphic signature scheme for triviality predicate* $\mathsf{pred}_{\mathsf{trivial}}$*. Then it is also* IND-CASA *indistinguishable for* $\mathsf{pred}_{\mathsf{trivial}}$*. In particular, for all PPT adversaries $\mathcal{A}$ that make at most q_D anamorphic decryption queries, there exist PPT adversaries $\mathcal{B}_0$ and $\mathcal{B}_1$ such that*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \leq 2q_D\, \mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{B}_0}(\lambda) + \mathbf{Adv}^{\mathsf{IND\text{-}CAMA}}_{\mathsf{aS},\mathcal{B}_1}(\lambda).$$

4.3 Dictator Forgeability of RRep, RIdP, and RIdPX

We now reexamine the security of the randomness replacement transform RRep (Figure 3) as well as the robust randomness identification with PRF transform RIdP and randomness identification with PRF and XOR transform RIdPX (Figure 4) in our dictator unforgeability notion. We find that all three are insecure due to “re-signing” attacks. The first two can be easily patched (to RRep* and RIdP*, specified in Section 3) while the third has no natural fix.

Dictator Forgeability of RRep. Consider RRep[S, prE], for which anamorphic signatures are computed via asig ← S.Sign(sk, msg; act) where act ← prE.Enc(dk, amsg). Note that, as specified in Construction 1, S must be randomness recovering for anamorphic decryption to be able to recover act.

The dictator can attack this scheme as follows. Suppose a user sends an anamorphic signature asig that contains a valid anamorphic message amsg and signs an innocuous message msg. The dictator, who knows the secret signing keys, can then extract act ← S.RRecov(vk, sk, msg, asig) and produce a *new* anamorphic signature by selecting a fresh innocuous msg* and signing asig* ← S.Sign(sk, msg*; act). The dictator sends along (msg*, asig*) as their forgery, and a recipient who runs RRep’s anamorphic decryption procedure first obtains act* ← S.RRecov(vk, sk, msg*, asig*) before computing amsg* ← prE.Dec(dk, act*). Observe that act* = act, so amsg* = amsg $\neq \bot$; hence, (msg*, asig*) is a valid forgery! From a broader perspective, the attack abuses the lack of “binding” between signing randomness act and innocuous message msg. The repaired RRep* transform simply adds this binding.

Dictator Forgeability of RIdP. Consider RIdP[S, prF], for which anamorphic signatures are computed via asig ← S.Sign(sk, msg; r) where r ← prF(dk, ($\mathsf{ctr}_{\mathsf{aSign}}$, amsg)). Furthermore, suppose S is randomness recovering. As shown in Table 3, many well-known signature schemes achieve this property.

The dictator can attack this scheme by leveraging the lack of binding between signing randomness r and innocuous message msg, as was done in the previous attack on RRep. Given an anamorphic signature asig that signs msg and hides a valid amsg, the dictator can extract r ← S.RRecov(vk, sk, msg, asig) and compute asig* ← S.Sign(sk, msg*; r) for a new msg*. As decryption computes r' ← prF(dk, ($\mathsf{ctr}_{\mathsf{aSign}}$, amsg$'$)) for all possible amsg$'$ and outputs the first that gives $r' = r$, anamorphic decryption outputs an amsg$^* \neq \bot$ and (msg*, asig*) is a valid forgery. The repaired RIdP* transform binds msg to r by including it as input to the pseudorandom function prF.

Dictator Forgeability of RIdPX. Consider $\mathsf{RIdPX}[\mathsf{S}, \mathsf{prF}]$, for which anamorphic signatures are computed via $\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$ where $r \leftarrow \mathsf{prF}(\mathsf{dk}, \mathsf{ctr}_{\mathsf{aSign}}) \oplus \mathsf{amsg}$. Let S be randomness recovering.

A dictator can perform the following attack, which is more potent than the attacks on RRep and RIdP. They begins by extracting $r \leftarrow \mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig})$ from an existing anamorphic signature asig that encrypts amsg. The dictator now computes $r^* \leftarrow r \oplus \mathsf{amsg} \oplus \mathsf{amsg}^* = \mathsf{prF}(\mathsf{dk}, \mathsf{ctr}_{\mathsf{aSign}}) \oplus \mathsf{amsg}^*$ and signs an innocuous message msg^* with sk to produce a signature asig^* which they submit as their forgery. This signature will decrypt to amsg^*.

Observe that straightforward methods to immunize RIdPX against this attack would likely involve providing amsg as input to prF. However, this approach would effectively convert RIdPX to RIdP, negating the efficiency advantages that BGHMR aimed to achieve with RIdPX.

4.4 Dictator Unforgeability of $\mathsf{RRep}^\star$ and $\mathsf{RIdP}^\star$

Our attacks on RRep, RIdP, and RIdPX highlight the need for non-malleable generation of signing randomness from the anamorphic message amsg that is bound to honest message msg. Repairing this binding, as we propose with the $\mathsf{RRep}^\star$ and $\mathsf{RIdP}^\star$ transforms, should result in dictator unforgeable anamorphic signature schemes. Although conceptually straightforward, formally proving general stateful results for requires certain technical details involving triviality predicates.

Restricted Classes of Triviality Predicates. For $\mathsf{aS} = \mathsf{RRep}^\star[\mathsf{S}, \mathsf{prE}]$ we would like to reduce dictator unforgeability ($\mathsf{DUF\text{-}CASA}$) to the ciphertext integrity ($\mathsf{INT\text{-}CT}$) of prE. Intuitively, since a reduction from $\mathsf{DUF\text{-}CASA}$ to $\mathsf{INT\text{-}CT}$ would simply forward all encryption/signing queries and decryption queries (after appropriate embedding/extraction), then the $\mathsf{DUF\text{-}CASA}$ and $\mathsf{INT\text{-}CT}$ notions should use the same triviality predicate $\mathsf{pred}_{\mathsf{trivial}}$. However, this approach does not hold in full generality, as $\mathsf{pred}_{\mathsf{trivial}}$ may exhibit entirely different behavior when applied to inputs from $\mathsf{S}.\mathbb{S}_{\mathsf{pp}}$ (i.e., signatures) compared to those from $\mathsf{prE}.\mathbb{C}_{\mathsf{pp}}$ (i.e., ciphertexts).

Bearing in mind the reduction from $\mathsf{DUF\text{-}CASA}$ to $\mathsf{INT\text{-}CT}$, it is clear that, while the query lists $\mathsf{L}_{\mathsf{aSign}}$ and $\mathsf{L}_{\mathsf{Enc}}$ maintained by each game contain different elements $(\mathsf{msg}_i, \mathsf{asig}_i)$ and $(\mathsf{ad}_i, \mathsf{ct}_i)$, there *is* a fixed correspondence due to the reduction's embedding and extraction procedures, that maps elements of one list to another. Thus, we want to narrow our focus to $\mathsf{pred}_{\mathsf{trivial}}$ that are concerned with the "equality patterns" of signatures/ciphertexts, rather than the signatures/ciphertexts themselves. If f is a function and $\mathsf{L} = [x_0, x_1, \ldots]$ is a list, let $f(\mathsf{L})$ denote the list obtained by replacing every x_i in L with $f(x_i)$ if $f(x_i) \neq \bot$ and removing it entirely otherwise.

Definition 13 (Equality-Pattern Respecting). *A triviality predicate* $\mathsf{pred}_{\mathsf{trivial}}$ *is* equality-pattern respecting *if* $\mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_1, \mathsf{L}_2) = \mathsf{pred}_{\mathsf{trivial}}(f(\mathsf{L}_1), f(\mathsf{L}_2))$ *for all lists* L_1 *and* L_2 *and all quasi-injections* f *(i.e., injections on inputs not mapped to* $\bot$*) such that* $f(x) \neq \bot$ *for all* $x \in \mathsf{L}_1$ *and* $x \in \mathsf{L}_2$*.*

When prE is permissive, we will additionally want the predicate to ignore elements of L_2 that are not in L_1. For a set $\mathbb{S}$, let $\mathsf{L} \setminus \mathbb{S}$ denote the list obtained by removing every x_i in L that is in $\mathbb{S}$.

Definition 14 (Permissiveness). *A triviality predicate* $\mathsf{pred}_{\mathsf{trivial}}$ *is* permissive *if* $\mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_1, \mathsf{L}_2) = \mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_1, \mathsf{L}_2 \setminus \mathbb{S})$ *for all lists* L_1 *and* L_2 *and all sets* $\mathbb{S}$ *such that* $x \notin \mathbb{S}$ *if* $x \in \mathsf{L}_1$ *or if* $x = \mathsf{L}_2[-1]$*.*

All of the predicates we define (e.g., in Section 2.2 and Appendix A.4) are equality-pattern respecting. All predicates we introduce as being for permissive schemes (e.g., $\mathsf{pred}^{\mathsf{perm}}_{\mathsf{trivial}}$, discussed in Section 2.2, and $\mathsf{pred}^{\mathsf{psync}}_{\mathsf{trivial}}$, defined shortly) are permissive.

Dictator Unforgeability of $\mathsf{RRep}^\star$. At a high level, the dictator unforgeability of $\mathsf{RRep}^\star$-derived anamorphic signature schemes directly follows from the ciphertext integrity of the underlying pseudorandom encryption scheme prE. This is because any attempt by the dictator to alter the signing randomness or msg will cause the authentication check of prE to fail, resulting in the signature scheme outputting $\bot$.

This intuition can be formalized by considering an adversary $\mathcal{A}$ against the dictator unforgeability of $\mathsf{aS} = \mathsf{RRep}^\star[\mathsf{S}, \mathsf{prE}]$, from which we construct a reduction $\mathcal{B}$ against the ciphertext integrity of prE. The reduction $\mathcal{B}$ simply runs signing operations internally (i.e., $\mathsf{S.KeyGen}$ and $\mathsf{S.Sign}$), and when $\mathcal{A}$ queries the anamorphic signing oracle O_{aSign} on input $(\mathsf{msg}, \mathsf{amsg})$, $\mathcal{B}$ obtains signing randomness act by querying its encryption oracle O_{Enc} on message amsg and associated data msg. Similarly, when $\mathcal{A}$ queries the anamorphic decryption oracle O_{aDec} on input $(\mathsf{msg}, \mathsf{asig})$, $\mathcal{B}$ uses $\mathsf{S.RRecov}$ to recover randomness act and sends this (along with associated data msg) to its decryption oracle O_{Dec}. When $\mathcal{A}$ outputs $(\mathsf{msg}^*, \mathsf{asig}^*)$, the reduction $\mathcal{B}$ uses $\mathsf{S.RRecov}$ to recover the randomness and outputs this as its forgery attempt. For permissive prE, the reduction $\mathcal{B}$ will refrain from forwarding decryption queries when randomness extraction fails, while for strict prE, the reduction $\mathcal{B}$ will additionally return $\perp$ to all future decryption queries.

We want to claim that $\mathcal{B}$ wins whenever $\mathcal{A}$ would. The crux of arguing this is to show that $\mathsf{pred}_{\mathsf{trivial}}$ responds "consistently" on the $(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}})$ that $\mathcal{A}$ expects its game to store and the $(\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}})$ that $\mathcal{B}$'s game stores. When prE and $\mathsf{pred}_{\mathsf{trivial}}$ are permissive, we know that dropping $(\mathsf{msg}, \mathsf{asig})$ when randomness extraction fails cannot change the output of the predicate. For strict schemes, if randomness extraction fails, then we no longer need to evaluate the predicate. Otherwise, we use the fact that $\mathsf{pred}_{\mathsf{trivial}}$ is equality-pattern respecting with the quasi-injection defined by $f((\mathsf{msg}, \mathsf{asig})) = (\mathsf{msg}, \mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig}))$ when $\mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig}) \neq \perp$ and $f((\mathsf{msg}, \mathsf{asig})) = \perp$ otherwise. The theorem follows from the reasoning above, with a full proof provided in Appendix C.2.

Theorem 3 ($\mathsf{RRep}^\star$ is $\mathsf{DUF\text{-}CASA}$). *Let S be a strongly randomness-recovering signature scheme and prE be an AEAD scheme achieving ciphertext integrity for equality-pattern-respecting triviality predicate $\mathsf{pred}_{\mathsf{trivial}}$ (which is permissive if prE is). Then $\mathsf{aS} = \mathsf{RRep}^\star[\mathsf{S}, \mathsf{prE}]$ (Construction 1) is dictator unforgeable for $\mathsf{pred}_{\mathsf{trivial}}$. In particular, for all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{INT\text{-}CT}}_{\mathsf{prE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{B}}(\lambda).$$

Note that if the randomness recovery of S is not strong, then an adversary may be able to create a forgery by mauling a $(\mathsf{msg}, \mathsf{asig})$ pair output by anamorphic signing into a different $(\mathsf{msg}, \mathsf{asig}^*)$ pair for which $\mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig}) = \mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig}^*)$.

We have already shown in Section 2 that the ElGamal signature scheme and RSA-PSS achieve strong randomness recovery. Thus, the above theorem implies that users in any network employing these signature schemes can select a suitable AEAD scheme and apply the $\mathsf{RRep}^\star$ transform to construct a dictator-unforgeable anamorphic signature scheme, allowing them to covertly communicate with strong confidentiality guarantees even in the presence of active dictators. Recall that for ElGamal, the signing key sk must be disclosed to anamorphic recipients (to allow them to extract signing randomness) and thus should only be used in settings where such recipients are trusted, whereas for RSA-PSS, this disclosure is unnecessary. The following corollaries capture these results.

Corollary 1 ($\mathsf{DUF\text{-}CASA}$ Symmetric Anamorphic ElGamal). *The ElGamal signature scheme can be made symmetric anamorphic and dictator unforgeable via $\mathsf{RRep}^\star$.*

Corollary 2 ($\mathsf{DUF\text{-}CASA}$ Non-Symmetric Anamorphic RSA-PSS). *RSA-PSS can be made non-symmetric anamorphic and dictator unforgeable via $\mathsf{RRep}^\star$.*

Users in networks implementing the ElGamal signature scheme who wish to covertly communicate with a dictator-unforgeable scheme, but do not trust anamorphic recipients with their signing keys, may turn to the $\mathsf{RIdP}^\star$ transform. It does not require signing key disclosure, and we now show it also produces dictator-unforgeable signature schemes.

Dictator Unforgeability of $\mathsf{RIdP}^\star$. We begin by analyzing the setting that motivated $\mathsf{RIdP}^\star$ to identify a triviality predicate under which it aims to be secure. Observe that $\mathsf{RIdP}^\star$ is both synchronous (in this case a sender and receiver maintain stateful counters $\mathsf{ctr}_{\mathsf{aSign}}$ and $\mathsf{ctr}_{\mathsf{aDec}}$) and permissive (Definition 7), meaning anamorphic decryption only succeeds on signatures in the order they were anamorphically signed, and that the decryption counter only updates when an anamorphic message is output. Thus, a forgery $(\mathsf{msg}^*, \mathsf{asig}^*)$ is trivial if the decryption counter $\mathsf{ctr}_{\mathsf{aDec}}$ is equal to the signing counter $\mathsf{ctr}_{\mathsf{aSign}}$ used to generate $(\mathsf{msg}^*, \mathsf{asig}^*)$, and non-trivial otherwise. This is captured by the *permissive synchronous* triviality predicate $\mathsf{pred}^{\mathsf{psync}}_{\mathsf{trivial}}$ shown in Figure 8, which functions for any appropriate scheme (e.g., encryption).

```
$\mathsf{pred}^{\mathsf{psync}}_{\mathsf{trivial}}(\mathsf{L}_1, \mathsf{L}_2)$
if $|\mathsf{L}_1| = 0 \vee |\mathsf{L}_2| = 0$ then
    return $0$
$\mathsf{ctr} \leftarrow 1$
for $1 \leq i \leq |\mathsf{L}_2| - 2$ do
    if $\mathsf{L}_1[\mathsf{ctr}] = \mathsf{L}_2[i]$ then
        $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
return $[\![\mathsf{L}_1[\mathsf{ctr}] = \mathsf{L}_2[-1]]\!]$
```

Fig. 8. Permissive synchronous triviality predicate $\mathsf{pred}^{\mathsf{psync}}_{\mathsf{trivial}}$

The following theorem formalizes the dictator unforgeability of $\mathsf{RIdP}^{\star}$-derived anamorphic signatures for $\mathsf{pred}^{\mathsf{psync}}_{\mathsf{trivial}}$. We provide defer the full proof to Appendix C.3.

Theorem 4 ($\mathsf{RIdP}^{\star}$ is $\mathsf{DUF\text{-}CASA}$). *Let* S *be a strongly randomness-identifying signature scheme and* prF *be pseudorandom function. Then* $\mathsf{aS} = \mathsf{RIdP}^{\star}[\mathsf{S}, \mathsf{prF}]$ *(Construction 2) is dictator unforgeable for* $\mathsf{pred}^{\mathsf{psync}}_{\mathsf{trivial}}$. *In particular, for all PPT adversaries $\mathcal{A}$ that make at most q_D anamorphic decryption queries and at most* $|\mathsf{aPKE}.\mathbb{ST}^{\mathsf{aEnc}}_{\mathsf{pp}}|$ *queries to either oracle, there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}^{\mathsf{psync}}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{B}}(\lambda) + (q_D + 1)\frac{|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|}{|\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}|}.$$

As noted in the preliminaries, writing $|\mathsf{aPKE}.\mathbb{ST}^{\mathsf{aEnc}}_{\mathsf{pp}}|$, $|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|$, and $|\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}|$ here is a slight abuse of notation as pp is only sampled during the execution of the security games. Formally, we require a polynomial upper bound on $|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|$ and a superpolynomial lower bound on $|\mathsf{aPKE}.\mathbb{ST}^{\mathsf{aEnc}}_{\mathsf{pp}}|$ and $|\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}|$.

Since we showed that ElGamal signatures are strongly randomness identifying, this result allows for non-symmetric anamorphic ElGamal.

Corollary 3 ($\mathsf{DUF\text{-}CASA}$ Non-Symmetric Anamorphic ElGamal). *The ElGamal signature scheme can be made non-symmetric anamorphic and dictator unforgeable via* $\mathsf{RIdP}^{\star}$.

The non-symmetric nature of this is less valuable than it may initially seem. In Section 5, we study the security that non-symmetric anamorphic signature schemes provide against malicious receivers. For ElGamal in particular, we observe in Section 5.7 that a receiver (knowing the "randomness" used for anamorphic signing) can use this to recover the secret signing key.

4.5 Extended Discussion and Results

Our motivation for dictator unforgeability centered around a gap between the robustness definition formalized by BGHMR and a security goal they outline, *not* that the high level goal of anamorphic robustness is without value. As an example of a use case of anamorphic robustness, one might define a robustness under chosen anamorphic message attack definition ($\mathsf{ROB\text{-}CAMA}$) that gives the adversary $(\mathsf{vk}, \mathsf{sk})$ and access to O_{aSign} and $O_{\mathsf{Sign},\mathsf{aDec}}$ where the latter either always returns $\perp$ or anamorphically decrypts honest signatures depending on a bit b. This alone does not guarantee security against active dictators but can be combined with integrity over several signatures to potentially achieve dictator unforgeability in the non-atomic setting. Here, a recipient would use the robustness check on each signature to identify the list of signatures fed into the integrity check.

A reader might have noticed that our definitions, proofs, and attacks regarding the transforms adapted from BGHMR (originally given for anamorphic encryption) were agnostic to the fact we were analyzing signature schemes. This is because the anamorphic setting considers a dictator who obtains both public

```
$\mathcal{G}^{\mathsf{PA\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$
$S \leftarrow \varnothing$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
$(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aSign}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow \mathsf{aS.aKeyGen}(\mathsf{pp})$
$(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}^{O_{\mathsf{Sign}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{dk})$
$b_{\mathsf{valid}} \leftarrow [\![\mathsf{aS.Vrfy}(\mathsf{vk}, \mathsf{msg}^*, \mathsf{sig}^*) = 1]\!]$
$b_{\mathsf{new}} \leftarrow [\![(\mathsf{msg}^*, \mathsf{sig}^*) \notin S]\!]$
return $b_{\mathsf{valid}} \wedge b_{\mathsf{new}}$
```

```
$O_{\mathsf{Sign}}(\mathsf{msg})$
$\mathsf{sig} \leftarrow \mathsf{aS.Sign}(\mathsf{sk}, \mathsf{msg})$
$S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{sig})\}$
return $\mathsf{sig}$
```

Fig. 9. Private anamorphism game $\mathcal{G}^{\mathsf{PA\text{-}CMA}}$ [KPP⁺23a]

and secret keys, so the distinction between using public or secret keys to produce ciphertexts versus signatures is irrelevant. Consequently, the results in this section also directly apply to the anamorphic encryption setting with minimal modifications. For completeness, we provide formal results on dictator unforgeability for anamorphic encryption in Appendix D, though we note that these are nearly identical to results given above.

5 Strengthening Private Anamorphism to Recipient Unforgeability

We now turn to a different security notion for anamorphic signatures. This notion, called private anamorphism, was proposed in KPPYZ [KPP$^+$23a] and codifies a desired form of security for non-symmetric anamorphic schemes (ones where a recipient does not obtain sk as part of dk). Motivated through our analysis of this security definition, we will propose a new unforgeability definition — this time from the perspective of a recipient rather than a dictator. To begin our analysis, we identify a gap in the formal definition of private anamorphism and present an attack against a scheme that satisfies this definition.

5.1 Private Anamorphism

Private anamorphism aims to allow two parties to covertly communicate via anamorphic signatures without the need to give up their signing keys. Otherwise a malicious anamorphic party could conceivably forge signatures on messages of their choosing for any one of their correspondents, and in the case where the double key is shared between multiple users, they may insert their own anamorphic messages as well.

The definition of private anamorphism formulated in KPPYZ requires that no PPT adversary with a plain signing oracle cannot produce a forgery on any message whatsoever, even when given both the verification and double keys. KPPYZ's notion of private anamorphism, which we denote as $\mathsf{PA\text{-}CMA}$, is captured by the $\mathcal{G}^{\mathsf{PA\text{-}CMA}}$ game shown in Figure 9. We define the $\mathsf{PA\text{-}CMA}$ advantage for an anamorphic signature scheme aS by

$$\mathbf{Adv}^{\mathsf{PA\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{PA\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \Rightarrow 1].$$

Definition 15 (Private Anamorphism). *An anamorphic signature scheme* aS *is private anamorphic (in the* $\mathsf{PA\text{-}CMA}$ *sense) if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{PA\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$ *is negligible.*

KPPYZ proved the following sufficiency result for private anamorphism.

Theorem 5 (Theorem 11 of [KPP$^+$23a]). *Let* aS *be an unforgeable anamorphic signature scheme where* $\mathsf{aS.aKeyGen}$ *is separable into the parallel and independent composition of algorithms that generate the signing keypair and double key. Then* aS *is private anamorphic. In particular, for all PPT adversaries* $\mathcal{A}$*, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{PA\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{aS},\mathcal{B}}(\lambda).$$

In the following subsection, we present a simple but practical attack against a scheme which satisfies Definition 15 and is hence private anamorphic, before proposing a new definition called recipient unforgeability which better captures the abilities and knowledge of parties using anamorphic signatures. We then revisit the randomness replacement transform RRep which forms the basis of many anamorphic signature schemes proposed in KPPYZ and show that it is susceptible to a similar practical attack. We rectify this by providing new sufficiency results for anamorphic signatures arrived at via $\mathsf{RRep}^\star$ (and hence RRep).

5.2 A Simple Attack

The attack we provide is contrived but nonetheless illustrates a gap between the definition of private anamorphism and reasonable abilities of an adversary. Suppose that Alice and Bob communicate using private anamorphic signatures, and Bob would like to steal Alice's signing key to impersonate her.

Alice and Bob use a bad private anamorphic signature scheme constructed from an existing private anamorphic signature scheme aS. The two operate identically, except for the bad anamorphic signing procedure which, given input anamorphic message amsg, outputs $\mathsf{asig} \leftarrow \mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} \| \mathsf{sk})$. Observe that the bad scheme is stealthy (Definition 8) as aS is itself stealthy and anamorphic signature schemes are stealthy regardless of what anamorphic messages are sent. Furthermore, it is indeed private anamorphic because its plain signing oracle is identical to that of aS; we have only changed anamorphic signing. However, it is trivially broken in anamorphic settings. When Alice sends a single anamorphic signature asig to Bob, he is able to obtain her signing key sk simply by decrypting asig.

5.3 Recipient Unforgeability

The attack discussed above relies on the fact both real *and* anamorphic signatures are sent. Recall the goal of private anamorphism: protecting the signature scheme integrity of a sender Alice against a malicious recipient Bob. In the course of their covert communication, Alice will send Bob many anamorphic messages hidden within signatures. In fact, Bob may even influence the messages Alice sends (such as asking her to perform computation he may have rigged).

As a result, we argue that a new security notion, which we call recipient unforgeability under chosen anamorphic message attack (RUF-CAMA), is necessary to replace private anamorphism. The difference of RUF-CAMA from the existing PA-CMA notion of private anamorphism is that the adversary receives access to both a real signing and anamorphic signing oracle. Let aS be an anamorphic signature scheme, then the $\mathcal{G}^{\mathsf{RUF\text{-}CAMA}}$ game is as shown in Figure 10. Unlike dictator unforgeability, we do not parameterize recipient unforgeability by a triviality predicate $\mathsf{pred}_{\mathsf{trivial}}$. This is because the forgery is with respect to the plain stateless signature scheme as opposed to the stateful anamorphic functionalities. The RUF-CAMA advantage for an anamorphic signature scheme aS is defined by

$$\mathbf{Adv}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \Rightarrow 1].$$

Definition 16 (Recipient Unforgeability). *An anamorphic signature scheme* aS *is recipient unforgeable (in the* RUF-CAMA *sense) if, for all PPT adversaries $\mathcal{A}$, the function $\mathbf{Adv}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$ is negligible.*

It is clear that recipient unforgeability for anamorphic signature schemes is a strictly stronger property than private anamorphism, since the adversary gets access to an additional oracle in the former over the latter. We formally capture this in the following theorem.

Theorem 6 (RUF-CAMA ⇒ PA-CMA). *Let* aS *be a recipient-unforgeable anamorphic signature scheme. Then, it is also private anamorphic. In particular, for all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{PA\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{B}}(\lambda).$$

5.4 Recipient Forgeability of RRep and $\mathsf{RRep}^\star$

Given the simple attack, we reexamine the randomness replacement transforms RRep and $\mathsf{RRep}^\star$ and find that they too are not necessarily secure in the envisioned deployment setting. As we will show, their weakness relies on the transmission of a chosen anamorphic message, which further motivates our proposed new recipient unforgeability definition.

```
$\mathcal{G}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$
    $S \leftarrow \varnothing$
    $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
    $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aSign}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow \mathsf{aS.aKeyGen}(\mathsf{pp})$
    $(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}^{O_{\mathsf{Sign}}, O_{\mathsf{aSign}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aDec}})$
    $b_{\mathsf{valid}} \leftarrow [\![\mathsf{aS.Vrfy}(\mathsf{vk}, \mathsf{msg}^*, \mathsf{sig}^*) = 1]\!]$
    $b_{\mathsf{new}} \leftarrow [\![(\mathsf{msg}^*, \mathsf{sig}^*) \notin S]\!]$
    return $b_{\mathsf{valid}} \wedge b_{\mathsf{new}}$

$O_{\mathsf{Sign}}(\mathsf{msg})$
    $\mathsf{sig} \leftarrow \mathsf{aS.Sign}(\mathsf{sk}, \mathsf{msg})$
    $S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{sig})\}$
    return $\mathsf{sig}$

$O_{\mathsf{aSign}}(\mathsf{msg}, \mathsf{amsg})$
    $\mathsf{asig} \leftarrow \mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aSign}})$
    $S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{asig})\}$
    return $\mathsf{asig}$
```

Fig. 10. Recipient unforgeability game $\mathcal{G}^{\mathsf{RUF\text{-}CAMA}}$

Recipient Forgeability of RRep. Like before, let Alice and Bob communicate using private anamorphic signatures, where Bob would like to steal Alice's signing key. This time Alice and Bob have chosen an anamorphic signature scheme arrived at via randomness replacement on a publicly randomness-recovering scheme (i.e., the signing randomness is replaced with pseudorandom encryptions of amsg, which Bob can extract and decrypt without sk). Alice believes this scheme protects her against malicious Bob as it is provably private anamorphic. However, the signature scheme contains a fatal flaw: when the randomness is equal to a magic value, the signing algorithm outputs its own secret key instead. Bob is then able to choose an anamorphic message that encrypts to the desired randomness and ask Alice to send it to him.

Stating the attack more concretely, let S be a publicly randomness-recovering signature scheme containing the weak point that signing with randomness equal to 0^λ outputs the secret key. Furthermore, let $\mathsf{prE}_{\mathsf{cm}}$ be a pseudorandom encryption scheme built from a λ-bit pseudorandom function prF used in counter mode. That is, $\mathsf{prE}_{\mathsf{cm}}.\mathsf{Enc}(\mathsf{k}, \mathsf{msg} : \mathsf{ctr}_{\mathsf{Enc}})$ outputs $\mathsf{prF}(\mathsf{k}, \mathsf{ctr}_{\mathsf{Enc}}) \oplus \mathsf{msg}$ and sets $\mathsf{ctr}_{\mathsf{Enc}} \leftarrow \mathsf{ctr}_{\mathsf{Enc}} + 1$ starting at $\mathsf{ctr}_{\mathsf{Enc}} = 0$. Consider the anamorphic signature scheme $\mathsf{RRep}[\mathsf{S}, \mathsf{prE}_{\mathsf{cm}}]$ (Construction 1). Straightforward reductions and the separable and independent key generation framework from KPPYZ show that aS^* is stealthy and private anamorphic.

Bob can obtain Alice's signing key by asking her to send him a message of his choosing. Suppose that Alice has already sent ℓ messages to Bob. Bob first assigns $\mathsf{ctr}^*_{\mathsf{Enc}} \leftarrow \ell + 1$, then, knowing k, computes $\mathsf{amsg}^* \leftarrow \mathsf{prF}(\mathsf{k}, \mathsf{ctr}^*_{\mathsf{Enc}}) \oplus (0^\lambda)$. Observe that $\mathsf{prE}_{\mathsf{cm}}.\mathsf{Enc}(\mathsf{dk}, \mathsf{amsg}^*) = \mathsf{prF}(\mathsf{k}, \ell{+}1) \oplus \mathsf{prF}(\mathsf{k}, \ell{+}1) \oplus (0^\lambda) = 0^\lambda$. Bob then asks Alice to send him amsg^*, triggering the weak point in the signature scheme allowing him to uncover her signing key.

Recipient Forgeability of RRep*. Recall that $\mathsf{RRep}^\star$ immunizes RRep against dictator forgeries by using an AEAD scheme which encrypts the anamorphic message amsg and authenticates both amsg and the honest message msg. Because this change only provides security guarantees against adversaries who don't know dk, Bob's attack on RRep also works on $\mathsf{RRep}^\star$ with appropriately backdoored primitives.

Let S be a publicly randomness-recovering signature scheme with signing key of length λ and signing randomness of length 2λ. Furthermore, let S have the weak point that signing with randomness r where the first λ bits are all 0 outputs the signing key. Now let $\mathsf{prE}_{\mathsf{auth\text{-}cm}}$ be an AEAD scheme with symmetric key $(\mathsf{k}_{\mathsf{Enc}}, \mathsf{k}_{\mathsf{MAC}})$ that outputs ciphertexts $\mathsf{ct} = \mathsf{ct}_{\mathsf{Enc}} \| \mathsf{ct}_{\mathsf{MAC}}$ where $\mathsf{ct}_{\mathsf{Enc}}$ is computed using $\mathsf{k}_{\mathsf{Enc}}$ with $\mathsf{prE}_{\mathsf{cm}}$ as defined above and $\mathsf{ct}_{\mathsf{MAC}} \leftarrow \mathsf{MAC}(\mathsf{k}_{\mathsf{MAC}}, \mathsf{ad} \| \mathsf{msg})$ for an appropriate MAC. For $\mathsf{RRep}^\star[\mathsf{S}, \mathsf{prE}_{\mathsf{auth\text{-}cm}}]$ (Construction 3), Bob again knows dk and can choose an amsg such that $\mathsf{ct}_{\mathsf{Enc}} = 0^\lambda$, exploiting the weak point in S regardless of the value of $\mathsf{ct}_{\mathsf{MAC}}$. Consequently, $\mathsf{RRep}^\star$ is also recipient unforgeable.

Bob's chosen message attack on both RRep and $\mathsf{RRep}^\star$ involves two sources of weakness: he exploits the symmetric scheme by controlling the structure of the output and he triggers an insecure point in the signature scheme. We discuss ways to mitigate either of these weaknesses in Sections 5.5 and 5.6. More broadly, with a new recipient unforgeability definition, the sufficiency results of KPPYZ for private anamorphism no longer apply to desired settings. We show that the stronger randomness replacement

```
$\mathcal{G}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{S},\mathcal{A}}(\lambda)$
    $S \leftarrow \varnothing$
    $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
    $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
    $(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}^{O_{\mathsf{Sign}}}(\mathsf{pp}, \mathsf{vk})$
    $b_{\mathsf{valid}} \leftarrow [\![\mathsf{S.Vrfy}(\mathsf{vk}, \mathsf{msg}^*, \mathsf{sig}^*) = 1]\!]$
    $b_{\mathsf{new}} \leftarrow [\![(\mathsf{msg}^*, \mathsf{sig}^*) \notin S]\!]$
    return $b_{\mathsf{valid}} \wedge b_{\mathsf{new}}$

$O_{\mathsf{Sign}}(\mathsf{msg}, r)$
    $\mathsf{sig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$
    $S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{sig})\}$
    return $\mathsf{sig}$
```

Fig. 11. Unforgeability under chosen randomness attack game $\mathcal{G}^{\mathsf{SUF\text{-}CRA}}$

transform $\mathsf{RRep}^\star$ (and consequently RRep) is still secure against malicious recipients, provided updated assumptions for the underlying primitives.

5.5 Recipient Unforgeability of $\mathsf{RRep}^\star$ with SUF-CRA-Secure Signatures

The chosen message attack on $\mathsf{RRep}^\star$ chiefly relies on the signature scheme's insecurity when randomness is chosen, as well as specific exploits of the pseudorandom encryption scheme by those who know the symmetric key. Here, we focus on the former by exploring signature schemes that are secure even when the signing randomness is chosen by an adversary. Consider the (strong) unforgeability game under chosen randomness attack game $\mathcal{G}^{\mathsf{SUF\text{-}CRA}}$ shown in Figure 11. The SUF-CRA advantage for a signature scheme S is defined as

$$\mathbf{Adv}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{S},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{S},\mathcal{A}}(\lambda) \Rightarrow 1].$$

Definition 17 (Chosen-Randomness Unforgeability). *A signature scheme* S *is* unforgeable under chosen randomness attack *(in the* SUF-CRA *sense) if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{S},\mathcal{A}}(\lambda)$ *is negligible.*

We now show that randomness replacement can construct recipient-unforgeable signatures from certain publicly-recovering schemes with minimal assumptions needed for the pseudorandom encryption scheme. In particular, we prove that the anamorphic signature scheme formed from $\mathsf{RRep}^\star[\mathsf{S}, \mathsf{prE}]$ is private anamorphic if S is SUF-CRA-secure and prE is pseudorandom. At a high level, this result follows from the fact that we can construct a reduction $\mathcal{B}$ against the SUF-CRA security of S that simulates the RUF-CAMA game to an adversary $\mathcal{A}$ by computing all prE operations ($\mathsf{prE.KeyGen}$ and $\mathsf{prE.Enc}$) internally to compute the relevant r, which it then queries (along with the corresponding benign message) to its signing oracle. Full details are in Appendix E.1.

Theorem 7 ($\mathsf{RRep}^\star$ with SUF-CRA Signatures is RUF-CAMA). *Let* S *be a chosen-randomness unforgeable publicly randomness-recovering signature scheme and* prE *be a pseudorandom encryption scheme. Then* $\mathsf{aS} = \mathsf{RRep}^\star[\mathsf{S}, \mathsf{prE}]$ *(Construction 1) is recipient unforgeable. In particular, for all PPT adversaries* $\mathcal{A}$*, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{S},\mathcal{B}}(\lambda).$$

We show that the anamorphic RSA-PSS variant (Figure 1) derived via $\mathsf{RRep}^\star$ is recipient unforgeable by showing that RSA-PSS is chosen-randomness unforgeable. Our proof bounds this by the hardness of RSA following an argument similar to the proof in [BR96]. The full proof is in Appendix E.2.

Lemma 1 (RSA-PSS is SUF-CRA). *RSA-PSS is chosen-randomness unforgeable, provided the hash functions* H *and* G *underlying it are modeled as random oracles and* λ_0 *is super-logarithmic in* λ*. In* *particular, for all PPT adversaries $\mathcal{A}$ that make up to q_S signing queries and q_H and q_G hash queries (to hash functions H and G respectively), there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{RSA\text{-}PSS},\mathcal{A}}(\lambda) \leq (q_S + q_H + 1)\left(2\,\mathbf{Adv}^{\mathsf{RSA}}_{\mathcal{B}}(\lambda) + (q_S + q_H + 1)2^{-\lambda-1} + (q_S + q_H + q_G + 1)2^{-\lambda_0}\right).$$

Corollary 4 (RUF-CAMA Anamorphic RSA-PSS). *RSA-PSS can be made anamorphic and recipient unforgeable via* $\mathsf{RRep}^{\star}$*.*

RSA-PSS was introduced as a variant of the RSA Full Domain Hash signature scheme [BR96] intended to achieve tight security. We were unable to preserve this tightness in the chosen-randomness setting, as we lost a multiplicative factor of $(q_S + q_H + 1)$.

5.6 Recipient Unforgeability of $\mathsf{RRep}^{\star}$ with SUF-CMA-Secure Signatures

We again recall that the chosen anamorphic message attack on RRep and $\mathsf{RRep}^{\star}$ (Section 5.4) leveraged both an insecurity in the underlying signature scheme and the pseudorandom encryption scheme. Having shown that signature schemes that satisfy a strong form of security can produce recipient unforgeable schemes via the $\mathsf{RRep}^{\star}$ transform, we now show that normal publicly randomness-recovering signature schemes can still achieve private anamorphism with the right choice of pseudorandom encryption scheme.

Intuitively, Bob's ability to mount the chosen anamorphic message attack relied on his ability to choose anamorphic messages that encrypt to ciphertexts of his choice. Typical security definitions for symmetric encryption schemes cannot mitigate these attacks. For instance, indistinguishability from random bits does not capture this kind of attack because it only considers an adversary who does not have access to the symmetric key. Instead, we desire that the ciphertext distribution in some way appears random, even to those who know the key. In the standard model such a property seems ill-defined because of course the distribution must depend on the key by the fact that decryption must be correct. However, this kind of property is achievable in ideal models. Let us illustrate the core idea with a simplified example.

We recall the example pseudorandom encryption scheme discussed in KPPYZ, which we refer to as randomized hash then XOR (RHtX). The scheme operates on λ-bit keys and messages and outputs 2λ-bit ciphertexts. Let $H : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$ be a hash function. Then $\mathsf{RHtX.Enc}(\mathsf{k}, \mathsf{msg})$ outputs $\mathsf{ct} \leftarrow r\|\gamma$ where r is a random λ-bit string and $\gamma \leftarrow \mathsf{msg} \oplus H(\mathsf{k}\|r)$.

Let $\mathsf{aS} = \mathsf{RRep}[\mathsf{S}, \mathsf{RHtX}]$. Then we claim aS is recipient unforgeable if S is unforgeable (and publicly randomness recovering), provided we model H as a random oracle which is not used by S. Given $\mathcal{A}$ against the recipient unforgeability of aS we can build a PPT $\mathcal{B}$ against the unforgeability of S. This $\mathcal{B}$ internally *simulates* H via *lazy sampling* and forwards non-anamorphic signing queries to its own signing oracle. Upon receiving an anamorphic signing query, $\mathcal{B}$ queries its own signing oracle to receive a signature, from which it extracts the randomness $\mathsf{act} = r\|\gamma$. It *programs* H so that $H(\mathsf{k}\|r) = \mathsf{msg} \oplus \gamma$ to ensure that RHtX decrypts properly. This reprogramming is undetectable unless $H(\mathsf{k}\|r)$ was already defined.

Note that RHtX is simply one fixed encryption scheme. For modularity, we wish to extract the core idea from above as a property to ask of (ideal model) encryption schemes. In practice, encryption schemes are built on block ciphers (possibly together with a hash function) which may be modeled as ideal ciphers, so we use a general syntax for an ideal primitive Prim (which we notate with syntax inspired by [Jae23,JT20]). The primitive Prim specifies the following three PPT algorithms: $\mathsf{Prim.Init}$ initializes the primitive's state, $\mathsf{Prim.LS}$ defines lazy sampling, and $\mathsf{Prim.Prog}$ defines programming. In security games, honest scheme algorithms use the primitive's lazy sampling procedure $\mathsf{Prim.LS}$ while simulators and adversaries additionally use $\mathsf{Prim.Prog}$.

The simulatability with random ciphertexts game $\mathcal{G}^{\mathsf{SIM\text{-}\$CT}}$ on a symmetric encryption scheme SE is as defined in Figure 12. In this game the adversary must distinguish between a real world where the adversary receives honest encryptions and a simulated one where SE calls a simulation algorithm $\mathsf{SE.Sim}$ which attempts to program Prim to be consistent with randomly sampled ciphertexts. The adversary additionally receives access to oracles $\mathsf{Prim.LS}$ and $\mathsf{Prim.Prog}$ for the underlying ideal primitive. Access to $\mathsf{Prim.Prog}$ is somewhat unnatural and is not essential to our results but is given for composability reasons

```
$\mathcal{G}^{\mathsf{SIM\text{-}\$CT}}_{\mathsf{SE},\mathsf{Prim},\mathcal{A}}(\lambda)$
  $b \leftarrow\!\!\$ \{0,1\}$
  $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
  $\mathsf{st} \leftarrow \mathsf{Prim.Init}(\mathsf{pp})$
  $(\mathsf{k}, \mathsf{st}_{\mathsf{Enc}}, \mathsf{st}_{\mathsf{Dec}}) \leftarrow \mathsf{SE.KeyGen}(\mathsf{pp})$
  $b^* \leftarrow \mathcal{A}^{O_{\mathsf{Enc}}, O_{\mathsf{LS}}, O_{\mathsf{Prog}}}(\mathsf{pp}, \mathsf{k}, \mathsf{st}_{\mathsf{Dec}})$
  return $[\![ b = b^* ]\!]$

$O_{\mathsf{Enc}}(\mathsf{ad}, \mathsf{msg})$
  if $b = 0$ then
    $\mathsf{ct} \leftarrow \mathsf{SE.Enc}^{O_{\mathsf{LS}}}(\mathsf{k}, \mathsf{ad}, \mathsf{msg} : \mathsf{st}_{\mathsf{Enc}})$
  else
    $\mathsf{ct} \leftarrow\!\!\$ \mathsf{SE}.\mathbb{C}_{\mathsf{pp}}$
    $\mathsf{SE.Sim}^{O_{\mathsf{LS}}, O_{\mathsf{Prog}}}(\mathsf{pp}, \mathsf{k}, \mathsf{ad}, \mathsf{msg}, \mathsf{ct} : \mathsf{st}_{\mathsf{Enc}})$
  return $\mathsf{ct}$

$O_{\mathsf{LS}}(x)$
  return $\mathsf{Prim.LS}(x : \mathsf{st})$

$O_{\mathsf{Prog}}(z)$
  return $\mathsf{Prim.Prog}(z : \mathsf{st})$
```

Fig. 12. Ciphertext simulatability game $\mathcal{G}^{\mathsf{SIM\text{-}\$CT}}$

identified by [CDG+18,Jae23]. We notate the SIM-\$CT advantage for a symmetric encryption scheme SE built on ideal primitive Prim with

$$\mathbf{Adv}^{\mathsf{SIM\text{-}\$CT}}_{\mathsf{SE},\mathsf{Prim},\mathcal{A}}(\lambda) = 2\Pr[\mathcal{G}^{\mathsf{SIM\text{-}\$CT}}_{\mathsf{SE},\mathsf{Prim},\mathcal{A}}(\lambda) \Rightarrow 1] - 1.$$

Definition 18 (Simulatability with Random Ciphertexts). *Let* SE *be a symmetric encryption scheme built on an ideal primitive* Prim*. It is* simulatable with random ciphertexts *(in the* SIM-\$CT *sense) if it specifies a PPT simulator* SE.Sim *such that, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{SIM\text{-}\$CT}}_{\mathsf{SE},\mathsf{Prim},\mathcal{A}}(\lambda)$ *is negligible.*

A full specification of RHtX in our generalized notation and security proof of SIM-\$CT is given in Appendix E.4. The key ideas in the proof extend to typical *randomized* modes of operation for block ciphers such as counter mode and cipher block chaining mode, though these modes exhibit ciphertext expansion due to the randomization which may be undesirable when trying to fit a ciphertext into signing randomness. Unfortunately, succinct symmetric encryption schemes are unrandomized, making them susceptible to the attack given in Section 5.4.

The following theorem captures how to construct private anamorphic signature schemes from normal publicly randomness-recovering signature schemes. At a high level, the theorem follows from a single game transition from a game $\mathcal{G}_0 = \mathcal{G}^{\mathsf{RUF\text{-}CAMA}(O_{\mathsf{LS}}, O_{\mathsf{Prog}})}$ on aS where each anamorphic signing query encrypts amsg to obtain act which is used as randomness during the signing process, to a hybrid $\mathcal{G}_1$, where act is instead randomly sampled and SE.Sim is run. Distinguishing $\mathcal{G}_0$ and $\mathcal{G}_1$ is bounded by the SIM-\$CT advantage for SE and Prim. Finally, given an $\mathcal{A}$ against $\mathcal{G}_1$, it is straightforward to construct a PPT $\mathcal{B}_1$ against the SUF-CMA security of S that responds to anamorphic signing queries by querying its own signing oracle, recovering the randomness act, and running SE.Sim. We provide a full proof in Appendix E.3.

Theorem 8 (RRep⋆ with SIM-\$CT Encryption is RUF-CAMA). *Let* S *be an unforgeable publicly randomness-recovering signature scheme and* prE *be a pseudorandom encryption scheme constructed from ideal primitive* Prim *that is simulatable with random ciphertexts. Then* $\mathsf{aS} = \mathsf{RRep}^\star[\mathsf{S}, \mathsf{prE}]$ *(Construction 1) is recipient unforgeable. In particular, for all PPT adversaries* $\mathcal{A}$*, there exist PPT adversaries* $\mathcal{B}_0$ *and* $\mathcal{B}_1$ *such that*

$$\mathbf{Adv}^{\mathsf{RUF\text{-}CAMA}(O_{\mathsf{LS}}, O_{\mathsf{Prog}})}_{\mathsf{aS},\mathsf{Prim},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{SIM\text{-}\$CT}}_{\mathsf{prE},\mathsf{Prim},\mathcal{B}_0}(\lambda) + \mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{S},\mathcal{B}_1}(\lambda).$$

S is independent of Prim so the last advantage function does not include it.

We briefly discuss anamorphic Boneh-Boyen signatures proposed in KPPYZ. The Boneh-Boyen signature scheme [BB08] operates over groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of order p and a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Signatures are of the form (r, s) where $r \leftarrow\!\!\$ \mathbb{Z}_p$ and s is derived from r and the signing key. The anamorphic

Boneh-Boyen signature scheme replaces r with a pseudorandom encryption of the anamorphic message (i.e., RRep). Since Boneh-Boyen is publicly randomness-recovering, anamorphic Boneh-Boyen is private anamorphic ($\mathsf{PA\text{-}CMA}$ secure).

Studying anamorphic Boneh-Boyen signatures in the context of our stronger proposed $\mathsf{RUF\text{-}CAMA}$ recipient unforgeability definition, we are unaware of any proof that plain Boneh-Boyen signatures are chosen randomness ($\mathsf{SUF\text{-}CRA}$) secure. This gap can be remedied applying our second result regarding $\mathsf{RRep}^\star$ because Boneh-Boyen signatures are $\mathsf{SUF\text{-}CMA}$ secure under a strong Diffie-Hellman assumption [BB08].

Corollary 5 (RUF-CAMA Anamorphic Boneh-Boyen). *The Boneh-Boyen signature scheme can be made anamorphic and recipient unforgeable via* $\mathsf{RRep}^\star$ *with* RHtX*.*

5.7 Dictator and Recipient (Un)Forgeable Schemes

Unless users are confident that a dictator is passive or that recipients can be fully trusted, those employing anamorphic signature schemes likely desire schemes that are both dictator unforgeable and recipient unforgeable. We conclude by analyzing which transform achieves both properties.

We begin by summarizing our positive results given in Sections 4 and 5. Per Theorem 3, Theorem 7, and Theorem 8, $\mathsf{RRep}^\star[\mathsf{S}, \mathsf{prE}]$ is both dictator unforgeable and recipient unforgeable provided S is publicly strongly randomness-recovering. Note that prE must be chosen depending on whether S is chosen-randomness unforgeable. Thus, users in networks utilizing RSA-PSS or Boneh-Boyen can stealthily communicate with comprehensive security.

Users could also use $\mathsf{RIdP}^\star$, designed to achieve *dictator* unforgeability per Theorem 4, to instantiate anamorphic channels through publicly randomness-identifying signature schemes including those mentioned above. As it turns out, the resulting anamorphic signature schemes are indeed *recipient* unforgeable provided the signature scheme is chosen-randomness secure. For completeness, we capture this in Theorem 9 with proof in Appendix E.5.

Theorem 9 ($\mathsf{RIdP}^\star$ with SUF-CRA Signatures is RUF-CAMA). *Let* S *be a chosen-randomness unforgeable signature scheme and* prF *be a pseudorandom function. Then* $\mathsf{aS} = \mathsf{RIdP}^\star[\mathsf{S}, \mathsf{prF}]$ *(Construction 2) is recipient unforgeable. In particular, for all PPT adversaries* $\mathcal{A}$*, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \le \mathbf{Adv}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{S},\mathcal{B}}(\lambda).$$

While $\mathsf{RIdP}^\star$ is dictator and recipient unforgeable for chosen-randomness secure signature schemes, we note that $\mathsf{RIdP}^\star$ is *not* recipient unforgeable for all anamorphic signature schemes *even though the signing key is never included in the double key.* Suppose Alice and Bob, as discussed in Corollary 3, communicate stealthily through ElGamal signatures using $\mathsf{RIdP}^\star$ which replaces signing randomness with $r \leftarrow \mathsf{prF}(\mathsf{dk}, (\mathsf{ctr}_{\mathsf{Enc}}, \mathsf{msg}, \mathsf{amsg}))$. When Alice sends a message msg and signature $\mathsf{asig} = (R, s)$ to Bob where $R \leftarrow g^r$ and $s \leftarrow (H(\mathsf{msg}, R) - xR)r^{-1} \pmod{p-1}$, Bob can obtain Alice's signing key by first anamorphically decrypting asig to retrieve amsg, computing $r \leftarrow \mathsf{prF}(\mathsf{dk}, (\mathsf{ctr}_{\mathsf{Dec}}, \mathsf{msg}, \mathsf{amsg}))$, then extracting $\mathsf{sk} = x \leftarrow (H(\mathsf{msg}, R) - rs)R^{-1} \pmod{p-1}$. This attack also works on $\mathsf{RIdP}^\star$ applied to DilithiumQROM [KLS18], a variant of ML-DSA, the new lattice-based NIST signature standard [Nat23b]. As DilithiumQROM signatures are of the form $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$, Bob can extract part of Alice's secret key $\mathbf{s}_1$ if all c are invertible in $R_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ (Lemma 2.2 of [LN17]). This holds as the coefficients of c are bounded and because $q \equiv 5 \pmod 8$ in all DilithiumQROM parameter sets.

Note that, per KPPYZ's separate and independent key generation framework (Theorem 5), $\mathsf{RIdP}^\star$ is private anamorphic. Hence, we have provided yet more examples of anamorphic constructions that satisfies their definition but is broken in practice. This attack transfers to RIdPX as well, thus we have provided reasonable instantiations of RIdPX that are *both* robust and private anamorphic, but *neither* dictator unforgeable (due to the attack we propose in Section 4.3) nor recipient unforgeable. That is, active dictators and malicious recipients could easily impersonate parties using RIdPX-derived anamorphic signature schemes even though RIdPX was (in essence) intended to be secure against both!

References

AMV20. Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. Subversion-resilient signatures: Definitions, constructions and applications. *Theoretical Computer Science*, 820:91–122, 2020. URL: https://www.sciencedirect.com/science/article/pii/S0304397520301808, doi:https://doi.org/10.1016/j.tcs.2020.03.021. 7

BB04. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73, Interlaken, Switzerland, May 2–6, 2004. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-24676-3_4. 10

BB08. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008. doi:10.1007/s00145-007-9005-7. 11, 31, 32

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, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-58723-8_1. 3, 4, 5, 6, 9, 14, 17, 18, 19, 20, 36, 39, 43, 48, 57, 61, 62, 63

BHMS16. Colin Boyd, Britta Hale, Stig Frode Mjølsnes, and Douglas Stebila. From stateless to stateful: Generic authentication and authenticated encryption constructions with application to TLS. In Kazue Sako, editor, *Topics in Cryptology – CT-RSA 2016*, volume 9610 of *Lecture Notes in Computer Science*, pages 55–71, San Francisco, CA, USA, February 29 – March 4, 2016. Springer, Cham, Switzerland. doi:10.1007/978-3-319-29485-8_4. 41

BJK15. Mihir Bellare, Joseph Jaeger, and Daniel Kane. Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 1431–1440, Denver, CO, USA, October 12–16, 2015. ACM Press. doi:10.1145/2810103.2813681. 7

BKN04. Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the encode-then-encrypt-and-mac paradigm. *ACM Trans. Inf. Syst. Secur.*, 7(2):206–241, 2004. doi:10.1145/996943.996945. 41

BPR14. Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 1–19, Santa Barbara, CA, USA, August 17–21, 2014. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-44371-2_1. 7

BR96. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416, Saragossa, Spain, May 12–16, 1996. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-68339-9_34. 10, 11, 29, 30

BR06. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer Berlin Heidelberg, Germany. doi:10.1007/11761679_25. 8

CCLZ25. Wonseok Choi, Daniel Collins, Xiangyu Liu, and Vassilis Zikas. A unified treatment of anamorphic encryption. Cryptology ePrint Archive, Report 2025/309, 2025. URL: https://eprint.iacr.org/2025/309. 8

CDG$^+$18. Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven. The wonderful world of global random oracles. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 280–312, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-78381-9_11. 31

CGM24. Dario Catalano, Emanuele Giunta, and Francesco Migliaro. Anamorphic encryption: New constructions and homomorphic realizations. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part II*, volume 14652 of *Lecture Notes in Computer Science*, pages 33–62, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-58723-8_2. 3, 8

Cor02. Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in

Computer Science, pages 272–287, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-46035-7_18. 11

Cra98. Scott Craver. On public-key steganography in the presence of an active warden. In David Aucsmith, editor, *Information Hiding*, pages 355–368, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. 7

DFP15. Jean Paul Degabriele, Pooya Farshim, and Bertram Poettering. A more cautious approach to security against mass surveillance. In Gregor Leander, editor, *Fast Software Encryption – FSE 2015*, volume 9054 of *Lecture Notes in Computer Science*, pages 579–598, Istanbul, Turkey, March 8–11, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-48116-5_28. 7

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, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland. doi:10.1007/978-3-032-01881-6_14. 7

ElG84. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Santa Barbara, CA, USA, August 19–23, 1984. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-39568-7_2. 10, 11

FGJ24. Marc Fischlin, Felix Günther, and Christian Janson. Robust channels: Handling unreliable networks in the record layers of QUIC and DTLS 1.3. *Journal of Cryptology*, 37(2):9, April 2024. doi:10.1007/s00145-023-09489-9. 39, 41

FGMP15. Marc Fischlin, Felix Günther, Giorgia Azzurra Marson, and Kenneth G. Paterson. Data is a stream: Security of stream-based channels. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 545–564, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-48000-7_27. 15

Jae23. Joseph Jaeger. Let attackers program ideal models: Modularity and composability for adaptive compromise. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 101–131, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-30620-4_4. 30, 31

JS18. Joseph Jaeger and Igors Stepanovs. Optimal channel security against fine-grained state compromise: The safety of messaging. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 33–62, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-96884-1_2. 39

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, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore. doi:10.1007/978-981-96-0888-1_4. 7, 21, 36

JT20. Joseph Jaeger and Nirvan Tyagi. Handling adaptive compromise for practical encryption schemes. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 3–32, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-56784-2_1. 30

KLS18. Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 552–586, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-78372-7_18. 32

KPB03. Tadayoshi Kohno, Adriana Palacio, and John Black. Building secure cryptographic transforms, or how to encrypt and MAC. Cryptology ePrint Archive, Report 2003/177, 2003. URL: https://eprint.iacr.org/2003/177. 41

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, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-38545-2_25. 3, 5, 6, 9, 11, 15, 16, 21, 26, 36, 37, 41

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, October 2023. doi:10.56553/popets-2023-0104. 3, 59, 60

LN17. Vadim Lyubashevsky and Gregory Neven. One-shot verifiable encryption from lattices. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 293–323, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-56620-7_11. 32

MS15. Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 657–686, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-46803-6_22. 7

Nat23a. National Institute of Standards. Digital signature standard (dss). Technical Report Federal Information Processing Standards Publication (FIPS) NIST FIPS 186-5, U.S. Department of Commerce, Washington, D.C., 2023. doi:https://doi.org/10.6028/NIST.FIPS.186-5. 11

Nat23b. National Institute of Standards. Module-lattice-based digital signature standard. Technical Report Federal Information Processing Standards Publication (FIPS) NIST FIPS 204 (Draft), U.S. Department of Commerce, Washington, D.C., 2023. 32

NY89. Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing*, pages 33–43, Seattle, WA, USA, May 15–17, 1989. ACM Press. doi:10.1145/73007.73011. 37

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, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-07085-3_2. 3, 14, 19, 57, 59

PS96. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398, Saragossa, Spain, May 12–16, 1996. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-68339-9_33. 11

RBBK01. Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001: 8th Conference on Computer and Communications Security*, pages 196–205, Philadelphia, PA, USA, November 5–8, 2001. ACM Press. doi:10.1145/501983.502011. 38

RZ18. Phillip Rogaway and Yusi Zhang. Simplifying game-based definitions - indistinguishability up to correctness and its application to stateful AE. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 3–32, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-96881-0_1. 41

Sch91. Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991. doi:10.1007/BF00196725. 10, 11

Sim83. Gustavus J. Simmons. The prisoners' problem and the subliminal channel. In David Chaum, editor, *Advances in Cryptology – CRYPTO'83*, pages 51–67, Santa Barbara, CA, USA, 1983. Plenum Press, New York, USA. doi:10.1007/978-1-4684-4730-9_5. 7

WCHY23. Yi Wang, Rongmao Chen, Xinyi Huang, and Moti Yung. Sender-anamorphic encryption reformulated: Achieving robust and generic constructions. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part VI*, volume 14443 of *Lecture Notes in Computer Science*, pages 135–167, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore. doi:10.1007/978-981-99-8736-8_5. 3

YY97. Adam Young and Moti Yung. Kleptography: Using cryptography against cryptography. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 62–74, Konstanz, Germany, May 11–15, 1997. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-69053-0_6. 7

YY04. Adam Young and Moti Yung. A subliminal channel in secret block ciphers. In Helena Handschuh and Anwar Hasan, editors, *SAC 2004: 11th Annual International Workshop on Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 198–211, Waterloo, Ontario, Canada, August 9–10, 2004. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-30564-4_14. 7

Changes From Proceedings Version

This article is based on an earlier article [JS24]. We refer to [JS24] as the "proceedings version" and this paper as the "full version". In addition to minor edits, organizational changes, and full proofs, the full version differs from the proceedings version as follows.

- In the proceedings version, we provide stateless definitions and theorems (with one exception). In the full version, all definitions and theorems are stateful.
- In the full version, we introduce *strong randomness recovery* and *strong randomness identification* properties not present in the proceedings version.
- In the proceedings version, we refer to signatures that need the signing key to recover randomness as *sk-randomness recovering* and ones that only need the verification key as *∅-randomness recovering*. In the full version, we call these *randomness recovering* and *publicly randomness recovering* respectively.
- In [BGH+24], the proposed robustness notion for anamorphic encryption does not give the public key to the adversary but allows the adversary to choose anamorphic decryption state. In the proceedings version, our definition of robustness ported to anamorphic signature schemes gives the public verification key to the adversary but does not allow the adversary to choose the anamorphic decryption state. In the full version, our ported definition of robustness matches the definition in [BGH+24].
- In the proceedings version, we provide a theorem that stateless anamorphic signature schemes that are dictator unforgeable are also robust. This theorem does not hold generically for stateful schemes, so we omit it in the full version and discuss why it does not hold.
- In the full version, we introduce CCA-style anamorphic message confidentiality and show that it is implied by dictator unforgeability and the CPA-style anamorphic message confidentiality notion proposed in [KPP+23a]. This definition and analysis are not present in the proceedings version.
- In [BGH+24], the RIdP and RIdPX transforms for anamorphic encryption (denoted Σ_1 and Σ_1' respectively in that work) don't input the honest message msg to the pseudorandom function. In the proceedings version, RIdP and RIdPX refer to transforms ported to anamorphic signatures that do input msg to the pseudorandom function. In the full version, RIdP and RIdPX match the constructions in [BGH+24]. We let RIdP* denote the RIdP transform specified in the proceedings version.
- In the proceedings version, we define RIdP and RIdPX as strict schemes. In the full version, we define them as permissive schemes, matching the constructions in [BGH+24].
- In the full version, we provide general dictator forgery attacks against RRep, RIdP, and RIdPX. In the proceedings version, we do not formally analyze RRep, we do not attack RIdP as we only specify the secure RIdP* transform (there, denoted RIdP), and our attack for RIdPX is less general.
- In the full version, we provide a strengthened RRep* transform that allows the pseudorandom encryption scheme to authenticate honest messages and show that it produces dictator unforgeable schemes. The transform and result are not present in the proceedings version.
- In the proceedings version, we prove the security of RIdP* (there, denoted RIdP) from a signature scheme property we introduced called *key unpredictability under key compromise*. This proof contains a small error fixed by introducing the strong randomness recovery and identification. In the full version, we omit key unpredictability and directly bound the security of RIdP*.
- In the full version, we provide anamorphic encryption versions of our dictator unforgeability results. These are not present in the proceedings version.
- In the proceedings version, we erroneously claim that chosen-randomness unforgeability of Rabin signatures tightly follows from the integer factorization assumption. Although Rabin signatures satisfy a non-strong variant of chosen-randomness unforgeability (implying non-strong variant of recipient unforgeability), we omit these results as they require additional definitions and proofs very similar to those already presented and are not central to our main contributions.
- In the proceedings version, we highlight RIdP* (there, denoted RIdP) as a scheme that achieves both dictator and recipient unforgeability when instantiated with publicly randomness-recovering signatures. In the full version, we highlight RRep* (only in the full version) as the transform that achieves this as it works on the same schemes and has a higher bandwidth than RIdP*.

```
$\mathcal{G}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{S},\mathcal{A}}(\lambda)$
$S \leftarrow \varnothing$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
$(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
$(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}^{O_{\mathsf{Sign}}}(\mathsf{pp}, \mathsf{vk})$
$b_{\mathsf{valid}} \leftarrow [\![\mathsf{S.Vrfy}(\mathsf{vk}, \mathsf{msg}^*, \mathsf{sig}^*) = 1]\!]$
$b_{\mathsf{new}} \leftarrow [\![(\mathsf{msg}^*, \mathsf{sig}^*) \notin S]\!]$
return $b_{\mathsf{valid}} \wedge b_{\mathsf{new}}$

$O_{\mathsf{Sign}}(\mathsf{msg})$
$\mathsf{sig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg})$
$S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{sig})\}$
return $\mathsf{sig}$
```

```
$\mathcal{G}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{A}}(\lambda)$
$b \leftarrow\!\!\$\ \{0, 1\}$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
$\mathsf{k} \leftarrow \mathsf{prF.KeyGen}(\mathsf{pp})$
$b^* \leftarrow \mathcal{A}^{O_{\mathsf{prF}}}(\mathsf{pp})$
return $[\![b = b^*]\!]$

$O_{\mathsf{prF}}(\mathsf{msg})$
if $\mathsf{T}[\mathsf{msg}] = \perp$ then
  if $b = 0$ then
    $\mathsf{T}[\mathsf{msg}] \leftarrow \mathsf{prF}(\mathsf{k}, \mathsf{msg})$
  else
    $\mathsf{T}[\mathsf{msg}] \leftarrow\!\!\$\ \mathsf{prF}.\mathbb{R}_{\mathsf{pp}}$
return $\mathsf{T}[\mathsf{msg}]$
```

Fig. 13. Signature scheme unforgeability game $\mathcal{G}^{\mathsf{SUF\text{-}CMA}}$ (left) and PRF game $\mathcal{G}^{\mathsf{PRF}}$ (right)

A Additional Notation and Preliminaries

We outline common cryptographic primitives we make use of through the paper, including signature schemes, public-key encryption, pseudorandom functions, and pseudorandom (symmetric) encryption. For each primitive, we formally define it, introduce relevant security definitions, and discuss concrete instantiations where applicable. In addition to syntax and security, also provide some general background discussion on modelling stateful encryption schemes.

A.1 Signature Schemes

Definition 19 (Signature Scheme). *A* signature scheme S *consists of three PPT algorithms.*

- $\mathsf{S.KeyGen}(\mathsf{pp})$ *takes public parameters* pp *and generates a signing key* $\mathsf{sk} \in \mathsf{S}.\mathbb{SK}_{\mathsf{pp}}$ *and verification key* $\mathsf{vk} \in \mathsf{S}.\mathbb{VK}_{\mathsf{pp}}$*. It outputs* $(\mathsf{vk}, \mathsf{sk})$.
- $\mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg})$ *takes a signing key* $\mathsf{sk} \in \mathsf{S}.\mathbb{SK}_{\mathsf{pp}}$ *and message* $\mathsf{msg} \in \mathsf{S}.\mathbb{M}_{\mathsf{pp}}$ *and outputs a signature* $\mathsf{sig} \in \mathsf{S}.\mathbb{S}_{\mathsf{pp}}$.
- $\mathsf{S.Vrfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$ *takes a verification key* $\mathsf{vk} \in \mathsf{S}.\mathbb{VK}_{\mathsf{pp}}$*, message* $\mathsf{msg} \in \mathsf{S}.\mathbb{M}_{\mathsf{pp}}$*, and signature* $\mathsf{sig} \in \mathsf{S}.\mathbb{S}_{\mathsf{pp}}$ *and outputs 1 if the signature is valid or 0 otherwise.*

Definition 20 (Correctness). *A signature scheme* S *is* correct *if* $\mathsf{S.Vrfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$ *outputs 1 for all* $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$*,* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$*,* $\mathsf{msg} \in \mathsf{S}.\mathbb{M}_{\mathsf{pp}}$*, and* $\mathsf{sig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg})$.

We consider (strong) unforgeability under chosen message attack. This notion is formalized in the $\mathcal{G}^{\mathsf{SUF\text{-}CMA}}$ game in Figure 13. The $\mathsf{SUF\text{-}CMA}$ advantage for a signature scheme S is defined as

$$\mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{S},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{S},\mathcal{A}}(\lambda) \Rightarrow 1].$$

Definition 21 (Unforgeability). *A signature scheme* S *is* unforgeable under chosen message attack (in the $\mathsf{SUF\text{-}CMA}$ sense) *if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{S},\mathcal{A}}(\lambda)$ *is negligible.*

We limit our analysis to stateless signature schemes. This choice simplifies notation and analysis while allowing us to cover most schemes. KPPYZ [KPP+23a] present only one stateful anamorphic signature scheme: a construction based on the Naor-Yung paradigm [NY89] which can be considered an instantiation of modified RRep transform applied to stateful signatures.

A.2 Public-Key Encryption Schemes

Definition 22 (Public-Key Encryption Scheme). *A* public-key encryption scheme PKE *consists of three PPT algorithms.*

- $\mathsf{PKE.KeyGen}(\mathsf{pp})$ *takes public parameters* pp *and generates a secret key* $\mathsf{sk} \in \mathsf{PKE}.\mathbb{SK}_{\mathsf{pp}}$ *and public key* $\mathsf{pk} \in \mathsf{PKE}.\mathbb{PK}_{\mathsf{pp}}$*. It outputs* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg})$ *takes a public key* $\mathsf{pk} \in \mathsf{PKE}.\mathbb{PK}_{\mathsf{pp}}$ *and message* $\mathsf{msg} \in \mathsf{PKE}.\mathbb{M}_{\mathsf{pp}}$ *and outputs a ciphertext* $\mathsf{ct} \in \mathsf{PKE}.\mathbb{C}_{\mathsf{pp}}$*.*
- $\mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{ct})$ *takes a secret key* $\mathsf{sk} \in \mathsf{PKE}.\mathbb{SK}_{\mathsf{pp}}$ *and ciphertext* $\mathsf{ct} \in \mathsf{PKE}.\mathbb{C}_{\mathsf{pp}}$ *and outputs a message* $\mathsf{msg} \in \mathsf{PKE}.\mathbb{M}_{\mathsf{pp}}$*.*

Definition 23 (Correctness). *A public-key encryption scheme* PKE *is* correct *if* $\mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{ct})$ *outputs* msg *for all* $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$*,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp})$*,* $\mathsf{msg} \in \mathsf{PKE}.\mathbb{M}_{\mathsf{pp}}$*, and* $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg})$*.*

Because we only consider public-key encryption in the anamorphic setting where a dictator obtains both public and secret keys, we don't consider any security notions for public-key encryption.

A.3 Pseudorandom Functions

Definition 24 (Pseudorandom Function). *A* pseudorandom function prF *consists of two PPT algorithms and must satisfy a security property we describe shortly.*

- $\mathsf{prF.KeyGen}(\mathsf{pp})$ *takes public parameters* pp *and outputs a symmetric key* $\mathsf{k} \in \mathsf{prF}.\mathbb{K}_{\mathsf{pp}}$*.*
- $\mathsf{prF}(\mathsf{k}, \mathsf{msg})$ *takes a key* $\mathsf{k} \in \mathsf{prF}.\mathbb{K}_{\mathsf{pp}}$ *and message* $\mathsf{msg} \in \mathsf{prF}.\mathbb{M}_{\mathsf{pp}}$ *and outputs a seed* $r \in \mathsf{prF}.\mathbb{R}_{\mathsf{pp}}$*.*

Pseudorandom functions must satisfy a pseudorandomness security notion formalized in the $\mathcal{G}^{\mathsf{PRF}}$ game in Figure 13. The PRF advantage for a pseudorandom function prF is defined as

$$\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{A}}(\lambda) = 2\Pr[\mathcal{G}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{A}}(\lambda) \Rightarrow 1] - 1.$$

For prF to be a pseudorandom function, it must hold that, for all PPT adversaries $\mathcal{A}$, the function $\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{A}}(\lambda)$ is negligible.

A.4 Pseudorandom Encryption

Pseudorandom (symmetric) encryption schemes encrypt messages into ciphertexts that appear uniformly random [RBBK01]. To formalize pseudorandom encryption, we first define (stateful) symmetric encryption schemes and then establish pseudorandomness as desired security property.

Definition 25 (Symmetric Encryption Scheme). *A* symmetric encryption scheme SE *consists of three PPT algorithms.*

- $\mathsf{SE.KeyGen}(\mathsf{pp})$ *takes public parameters* pp *and generates a symmetric key* $\mathsf{k} \in \mathsf{SE}.\mathbb{K}_{\mathsf{pp}}$*, initial encryption state* $\mathsf{st}_{\mathsf{Enc}} \in \mathsf{SE}.\mathbb{ST}^{\mathsf{Enc}}_{\mathsf{pp}}$*, and initial decryption state* $\mathsf{st}_{\mathsf{Dec}} \in \mathsf{SE}.\mathbb{ST}^{\mathsf{Dec}}_{\mathsf{pp}}$*. It outputs* $(\mathsf{k}, \mathsf{st}_{\mathsf{Enc}}, \mathsf{st}_{\mathsf{Dec}})$*.*
- $\mathsf{SE.Enc}(\mathsf{k}, \mathsf{ad}, \mathsf{msg} : \mathsf{st}_{\mathsf{Enc}})$ *takes a symmetric key* $\mathsf{k} \in \mathsf{SE}.\mathbb{K}_{\mathsf{pp}}$*, associated data* $\mathsf{ad} \in \mathsf{SE}.\mathbb{AD}_{\mathsf{pp}}$*, message* $\mathsf{msg} \in \mathsf{SE}.\mathbb{M}_{\mathsf{pp}}$*, and encryption state* $\mathsf{st}_{\mathsf{Enc}} \in \mathsf{SE}.\mathbb{ST}^{\mathsf{Enc}}_{\mathsf{pp}}$ *and outputs a ciphertext* $\mathsf{ct} \in \mathsf{SE}.\mathbb{C}_{\mathsf{pp}}$*. It also updates the state* $\mathsf{st}_{\mathsf{Enc}}$*.*
- $\mathsf{SE.Dec}(\mathsf{k}, \mathsf{ad}, \mathsf{ct} : \mathsf{st}_{\mathsf{Dec}})$ *takes a key* $\mathsf{k} \in \mathsf{SE}.\mathbb{K}_{\mathsf{pp}}$*, associated data* $\mathsf{ad} \in \mathsf{SE}.\mathbb{AD}_{\mathsf{pp}}$*, ciphertext* $\mathsf{ct} \in \mathsf{SE}.\mathbb{C}_{\mathsf{pp}}$*, and state* $\mathsf{st}_{\mathsf{Dec}} \in \mathsf{SE}.\mathbb{ST}^{\mathsf{Dec}}_{\mathsf{pp}}$ *and outputs a message* $\mathsf{msg} \in \mathsf{SE}.\mathbb{M}_{\mathsf{pp}} \cup \{\bot\}$*. It also updates the state* $\mathsf{st}_{\mathsf{Dec}}$*.*

```
$\mathcal{G}^{\mathsf{CORR}}_{\mathsf{SE},\mathcal{A}}(\lambda)$
$\mathsf{win} \leftarrow 0$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
$(\mathsf{k}, \mathsf{st}_{\mathsf{Enc}}, \mathsf{st}_{\mathsf{Dec}}) \leftarrow \mathsf{SE.KeyGen}(\mathsf{pp})$
$\textbf{run } \mathcal{A}^{O_{\mathsf{Enc},\mathsf{Dec}}}(\mathsf{pp})$
$\textbf{return } \mathsf{win}$
```

```
$O_{\mathsf{Enc},\mathsf{Dec}}(\mathsf{ad}, \mathsf{msg})$
$\mathsf{ct} \leftarrow \mathsf{SE.Enc}(\mathsf{k}, \mathsf{ad}, \mathsf{msg} : \mathsf{st}_{\mathsf{Enc}})$
$\mathsf{msg}^* \leftarrow \mathsf{SE.Dec}(\mathsf{k}, \mathsf{ad}, \mathsf{ct} : \mathsf{st}_{\mathsf{Dec}})$
$\mathsf{win} \leftarrow \mathsf{win} \vee [\![\mathsf{msg} \neq \mathsf{msg}^*]\!]$
$\textbf{return } \mathsf{ct}$
```

```
$\mathcal{G}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{SE},\mathcal{A}}(\lambda)$
$b \leftarrow\!\!\$\ \{0, 1\}$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
$(\mathsf{k}, \mathsf{st}_{\mathsf{Enc}}, \mathsf{st}_{\mathsf{Dec}}) \leftarrow \mathsf{SE.KeyGen}(\mathsf{pp})$
$b^* \leftarrow \mathcal{A}^{O_{\mathsf{Enc}}}(\mathsf{pp})$
$\textbf{return } [\![b = b^*]\!]$
```

```
$O_{\mathsf{Enc}}(\mathsf{ad}, \mathsf{msg})$
$\textbf{if } b = 0 \textbf{ then}$
    $\mathsf{ct} \leftarrow \mathsf{SE.Enc}(\mathsf{k}, \mathsf{ad}, \mathsf{msg} : \mathsf{st}_{\mathsf{Enc}})$
$\textbf{else}$
    $\mathsf{ct} \leftarrow\!\!\$\ \mathsf{SE}.\mathbb{C}_{\mathsf{pp}}$
$\textbf{return } \mathsf{ct}$
```

Fig. 14. Symmetric encryption correctness game $\mathcal{G}^{\mathsf{CORR}}$ (top) and pseudorandomness game $\mathcal{G}^{\mathsf{IND\$\text{-}CPA}}$ (bottom)

We will omit the states as inputs or outputs to algorithms when SE is stateless. Some results hold regardless of whether SE does or does not provide authentication. In these cases, we provide associated data as input to algorithms which may ignore it. We will sometimes omit the associated data as inputs to algorithms for results or constructions in which SE never performs authentication.

We leverage a game-based approach to define a minimal notion of correctness for symmetric encryption schemes, which we formalize in Figure 14 as the $\mathcal{G}^{\mathsf{CORR}}$ game. The CORR advantage for a pseudorandom encryption scheme prE is defined as

$$\mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{prE},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{CORR}}_{\mathsf{prE},\mathcal{A}}(\lambda) \Rightarrow 1].$$

Definition 26 (Correctness). *A pseudorandom encryption scheme* prE *is* correct *if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{prE},\mathcal{A}}(\lambda)$ *is negligible.*

This matches the most basic version of correctness property that any stateful symmetric encryption scheme should achieve: ciphertexts should decrypt properly if they are delivered in order without modification (with corresponding associated data). In different settings, one might want stronger correctness properties requiring that ciphertexts decrypt properly even when delivered in some other order. As mentioned in Sections 2 and 3, it is useful to distinguish between strict schemes which will permanently reject all future ciphertexts if they ever reject a ciphertext, and permissive schemes that will act like they never saw a rejected ciphertext.

Definition 27 (Strictness/Permissiveness). *A symmetric encryption scheme* SE *is* permissive *if* $\mathsf{st}_{\mathsf{Dec}}$ *is unmodified by* $\mathsf{msg} \leftarrow \mathsf{SE.Dec}(\mathsf{k}, \mathsf{ad}, \mathsf{ct} : \mathsf{st}_{\mathsf{Dec}})$ *whenever* $\mathsf{msg} = \bot$*. A symmetric encryption scheme* SE *is* strict *if* $\mathsf{st}_{\mathsf{Dec}}$ *is set to* $\bot$ *in the same circumstance and* $\mathsf{SE.Dec}(\mathsf{k}, \mathsf{ad}, \mathsf{ct} : \bot)$ *always outputs* $\bot$*.*

Permissive schemes are sometimes called robust [FGJ24,JS18]. We avoid this terminology because it overlaps with the robustness security notion proposed by BGHMR [BGH$^+$24] that we reanalyze in this work. Note, however, that these two notions of robustness share the common feature that they are only meaningful to consider for permissive schemes.

Pseudorandomness. To capture pseudorandomness for symmetric encryption schemes, we define the indistinguishability from random under chosen plaintext attack game $\mathcal{G}^{\mathsf{IND\$\text{-}CPA}}$ on a symmetric encryption scheme SE in Figure 14. Briefly, the game models an adversary's ability to distinguish valid ciphertexts

```
$\mathcal{G}^{\mathsf{INT\text{-}CT}}_{\mathsf{SE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda)$
$(\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}}) \leftarrow ([\cdot], [\cdot])$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^{\lambda})$
$(\mathsf{k}, \mathsf{st}_{\mathsf{Enc}}, \mathsf{st}_{\mathsf{Dec}}) \leftarrow \mathsf{SE.KeyGen}(\mathsf{pp})$
$(\mathsf{ad}^*, \mathsf{ct}^*) \leftarrow \mathcal{A}^{O_{\mathsf{Enc}}, O_{\mathsf{Dec}}}(\mathsf{pp})$
$\mathsf{L}_{\mathsf{Dec}}.\mathsf{add}((\mathsf{ad}^*, \mathsf{ct}^*))$
$b_{\mathsf{valid}} \leftarrow [\![\mathsf{SE.Dec}(\mathsf{k}, \mathsf{ad}^*, \mathsf{ct}^* : \mathsf{st}_{\mathsf{Dec}}) \neq \bot]\!]$
$b_{\mathsf{trivial}} \leftarrow \mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}})$
return $b_{\mathsf{valid}} \wedge \neg b_{\mathsf{trivial}}$
```

```
$O_{\mathsf{Enc}}(\mathsf{ad}, \mathsf{msg})$
$\mathsf{ct} \leftarrow \mathsf{SE.Enc}(\mathsf{k}, \mathsf{ad}, \mathsf{msg} : \mathsf{st}_{\mathsf{Enc}})$
$\mathsf{L}_{\mathsf{Enc}}.\mathsf{add}((\mathsf{ad}, \mathsf{ct}))$
return $\mathsf{ct}$
```

```
$O_{\mathsf{Dec}}(\mathsf{ad}, \mathsf{ct})$
$\mathsf{msg} \leftarrow \mathsf{SE.Dec}(\mathsf{k}, \mathsf{ad}, \mathsf{ct} : \mathsf{st}_{\mathsf{Dec}})$
$\mathsf{L}_{\mathsf{Dec}}.\mathsf{add}((\mathsf{ad}, \mathsf{ct}))$
return $\mathsf{msg}$
```

```
$\mathsf{pred}_{1s}(\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}})$
return $[\![\forall 1 \leq j \leq |\mathsf{L}_{\mathsf{Dec}}|, \exists i : \mathsf{L}_{\mathsf{Enc}}[i] = \mathsf{L}_{\mathsf{Dec}}[j]]\!]$
```

```
$\mathsf{pred}_{1p}(\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}})$
return $[\![\exists i : \mathsf{L}_{\mathsf{Enc}}[i] = \mathsf{L}_{\mathsf{Dec}}[-1]]\!]$
```

```
$\mathsf{pred}_{2s}(\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}})$
return $[\![\forall 1 \leq i \leq |\mathsf{L}_{\mathsf{Dec}}| : \mathsf{L}_{\mathsf{Enc}}[i] = \mathsf{L}_{\mathsf{Dec}}[i]]\!]$
```

```
$\mathsf{pred}_{2p}(\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}})$
$i, j \leftarrow 1$
while $i \leq |\mathsf{L}_{\mathsf{Enc}}| \wedge j \leq |\mathsf{L}_{\mathsf{Dec}}|$ do
  if $\mathsf{L}_{\mathsf{Enc}}[i] = \mathsf{L}_{\mathsf{Dec}}[j]$ do
    if $j = |\mathsf{L}_{\mathsf{Dec}}|$ then return $1$
    $i \leftarrow i + 1$
  $j \leftarrow j + 1$
return $0$
```

Fig. 15. Ciphertext integrity game $\mathcal{G}^{\mathsf{INT\text{-}CT}}$ (top) and example triviality predicates (bottom)

from fresh random samples from the ciphertext-space. Note that our syntax implicitly assumes we are only considering fixed-size messages, so the ciphertext-space $\mathsf{SE}.\mathbb{C}_{\mathsf{pp}}$ does not depend on the message. The $\mathsf{IND\$\text{-}CPA}$ advantage for a symmetric encryption scheme SE is defined as

$$\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{SE},\mathcal{A}}(\lambda) = 2\Pr[\mathcal{G}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{SE},\mathcal{A}}(\lambda) \Rightarrow 1] - 1.$$

Definition 28 (Pseudorandomness). *A stateful symmetric encryption scheme* SE *is a* pseudorandom encryption scheme *if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{SE},\mathcal{A}}(\lambda)$ *is negligible.*

We write prE to denote pseudorandom encryption schemes.

Encryption Unforgeability. To consider the unforgeability of a symmetric encryption scheme, we define the ciphertext integrity game $\mathcal{G}^{\mathsf{INT\text{-}CT}}$ on a symmetric encryption scheme SE in Figure 15. Briefly, the game models an adversary's ability to forge a new ciphertext that decrypts to a non-$\bot$ value after being given example encryptions on messages of its choice. The $\mathsf{INT\text{-}CT}$ advantage for a symmetric encryption scheme SE and triviality predicate $\mathsf{pred}_{\mathsf{trivial}}$ is defined as

$$\mathbf{Adv}^{\mathsf{INT\text{-}CT}}_{\mathsf{SE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{INT\text{-}CT}}_{\mathsf{SE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \Rightarrow 1].$$

Definition 29 (Unforgeability/Ciphertext Integrity). *A stateful symmetric encryption scheme* SE *is unforgeable (in the* $\mathsf{INT\text{-}CT}$ *sense) for a triviality predicate* $\mathsf{pred}_{\mathsf{trivial}}$ *if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{INT\text{-}CT}}_{\mathsf{SE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda)$ *is negligible.*

This definition is parameterized by a predicate $\mathsf{pred}_{\mathsf{trivial}}$ (first discussed in Section 2) to capture that different deployment scenarios may call for encryption schemes providing different levels of protection against, e.g., forging, replaying, reordering, or dropping messages. The decisions of what properties to aim for tend to depend on the reliability of the underlying mechanism by which ciphertexts are delivered from sender to receiver (for example, TLS, which is designed for the reliable transport protocol TCP, makes different design decisions than DTLS and QUIC, which are designed for unreliable transport). In this work, we aim to be agnostic of the underlying transport mechanism, because the precise deployment scenario of anamorphic cryptography is unclear, and so give this parameterized definition.

Four examples are given in Figure 15. Predicate pred_{1s} considers ciphertexts (with corresponding associated data) trivial only if every ciphertext received by decryption so far was previously returned by encryption. Predicate pred_{2s} considers ciphertexts trivial only if every ciphertext received by decryption was previously returned by encryption *in the exact same order*. In the body, it is referred to as $\mathsf{pred}^{\mathsf{sync}}_{\mathsf{trivial}}$. These predicates are reasonable for use with strict encryption schemes.

Predicates pred_{1p} and pred_{2p} are the natural variants of pred_{1s} and pred_{2s} for permissive schemes. They declare the ciphertexts trivial under "analogous" conditions as the strict predicates, except ignoring elements of $\mathsf{L}_{\mathsf{Dec}}$ that would have be considered non-trivial. In the body, they are referred to as $\mathsf{pred}^{\mathsf{perm}}_{\mathsf{trivial}}$ and $\mathsf{pred}^{\mathsf{psync}}_{\mathsf{trivial}}$, respectively. If a ciphertext is successfully forged (i.e., accepted when declared non-trivial) against pred_{2p}, future ciphertexts may erroneously be declared trivial forgeries.

For $x \in \{1, 2\}$, one can easily switch between strict stateful schemes secure with pred_{xs} and permissive schemes with pred_{xp} by modifying the behavior of decryption. Stateless schemes can only achieve security with respect to pred_{1p}. For further discussions of the different choices for defining unforgeability (and the corresponding notions of correctness) of stateful encryption schemes, we refer the reader to [BKN04,BHMS16,FGJ24,KPB03,RZ18] and related work.

B Details on Constructing Anamorphic Signatures

In this section, we show that the previously proposed anamorphic constructions we revisit in this work (RRep, RIdP, and RIdPX) and our new constructions (RRep* and RIdP*) achieve correctness and the established stealthiness and robustness notions if applicable. In the case of the previous constructions, we provide these results because we analyze a construction that captures the core idea behind several existing constructions (in the case of RRep) or because our results are for the same construction on a different underlying primitive (in the case RIdP and RIdPX).

B.1 Details on RRep and RRep*

Details on RRep. We provide theorem statements of the correctness and stealthiness (in the RoA-CAMA sense) of RRep. The proofs of these theorems follow from later security proofs for RRep*, which is strict generalization of RRep. We don't consider robustness because RRep was not designed to achieve it. We begin with correctness. Note that KPPYZ [KPP$^+$23a] consider only perfectly correct stateless schemes, while our result below considers a form of stateful imperfect correctness as well.

Theorem 10 (RRep is Correct). *Let* S *be a randomness-recovering signature scheme and* prE *be a pseudorandom encryption scheme. Then* $\mathsf{aS} = \mathsf{RRep}[\mathsf{S}, \mathsf{prE}]$ *(Construction 1) is correct. In particular, for all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{prE},\mathcal{B}}(\lambda).$$

We now move to stealthiness of RRep, which we capture in the theorem below. This result is essentially the combination of Theorems 9 and 10 of KPPYZ [KPP$^+$23a], which state that the Fiat-Shamir transform applied to randomness-recovering identification protocols produce anamorphic signature schemes. Note, however, that this result applies to randomness recovering signature schemes regardless of whether or not they are derived via the Fiat-Shamir transform, though it does not apply to Fiat-Shamir-derived anamorphic signatures where the underlying identification protocol doesn't use randomness replacement.

Theorem 11 (RRep is RoA-CAMA). *Let* S *be a randomness-recovering signature scheme and* prE *be a pseudorandom encryption scheme. Then* $\mathsf{aS} = \mathsf{RRep}[\mathsf{S}, \mathsf{prE}]$ *(Construction 1) is stealthy. In particular, for all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{prE},\mathcal{B}}(\lambda).$$

```
$\mathcal{B}^{O^{\mathcal{B}}_{\mathsf{Enc,Dec}}}(\mathsf{pp})$
    $(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
    run $\mathcal{A}^{O^{\mathcal{A}}_{\mathsf{aSign,aDec}}}(\mathsf{pp},\mathsf{vk},\mathsf{sk})$

$O^{\mathcal{A}}_{\mathsf{aSign,aDec}}(\mathsf{msg},\mathsf{amsg})$
    $\mathsf{act} \leftarrow O^{\mathcal{B}}_{\mathsf{Enc,aDec}}(\mathsf{msg},\mathsf{amsg})$
    $\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk},\mathsf{msg};\mathsf{act})$
    return $\mathsf{asig}$
```

```
$\mathcal{B}^{O^{\mathcal{B}}_{\mathsf{Enc}}}(\mathsf{pp})$
    $(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
    $b^* \leftarrow \mathcal{A}^{O^{\mathcal{A}}_{\mathsf{Sign/aSign}}}(\mathsf{pp},\mathsf{vk},\mathsf{sk})$
    return $\neg b^*$

$O^{\mathcal{A}}_{\mathsf{Sign/aSign}}(\mathsf{msg},\mathsf{amsg})$
    $\mathsf{act} \leftarrow O^{\mathcal{B}}_{\mathsf{Enc}}(\mathsf{msg},\mathsf{amsg})$
    $\mathsf{asig} \leftarrow \mathsf{aS.Sign}(\mathsf{sk},\mathsf{msg};\mathsf{act})$
    return $\mathsf{asig}$
```

Fig. 16. Reductions used in the proofs of Theorem 12 (top) and Theorem 13 (bottom)

Details on $\mathsf{RRep}^\star$. We formally establish the correctness and stealthiness of our proposed $\mathsf{RRep}^\star$ transform in the following theorems.

Theorem 12 ($\mathsf{RRep}^\star$ is Correct). *Let S be a randomness-recovering signature scheme and prE be a pseudorandom encryption scheme. Then $\mathsf{aS} = \mathsf{RRep}^\star[\mathsf{S},\mathsf{prE}]$ (Construction 1) is correct. In particular, for all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{prE},\mathcal{B}}(\lambda).$$

Proof. At a high level, the theorem follows from the fact that $\mathsf{RRep}^\star$, which replaces signing randomness with pseudorandom encryptions, requires that the underlying signature scheme S can recover randomness perfectly. Hence, any anamorphic signature that breaks correctness must break the correctness of the underlying pseudorandom encryption scheme prE. This intuition is captured by constructing a PPT reduction $\mathcal{B}$ against the correctness of prE from a PPT adversary against the correctness of aS.

We first review the games that $\mathcal{A}$ and $\mathcal{B}$ play. The adversary $\mathcal{A}$ plays the $\mathcal{G}^{\mathsf{CORR}}$ game for anamorphic signatures where it receives the signing keypair $(\mathsf{vk},\mathsf{sk})$ and has access to an anamorphic-sign-then-anamorphic-decrypt oracle $O^{\mathcal{A}}_{\mathsf{aSign,aDec}}$ to which it can query tuples $(\mathsf{msg},\mathsf{amsg})$ and receives an anamorphic signature asig generated using an anamorphic signing state $\mathsf{st}_{\mathsf{aSign}}$ that is then updated. It wins if the anamorphic decryption of asig, when supplied with msg and an anamorphic decryption state $\mathsf{st}_{\mathsf{aDec}}$ that is also updated, is not amsg for any query to this oracle. The reduction $\mathcal{B}$ plays the $\mathcal{G}^{\mathsf{CORR}}$ game for symmetric encryption where it accesses an encrypt-then-decrypt oracle $O^{\mathcal{B}}_{\mathsf{Enc,Dec}}$ to which it can query messages msg and associated data ad and receive ciphertexts ct encrypted with an updating encryption state $\mathsf{st}_{\mathsf{Enc}}$. It wins if the decryption of ct using ad and an updating decryption state $\mathsf{st}_{\mathsf{Dec}}$ is not msg.

We construct $\mathcal{B}$ from $\mathcal{A}$ as shown in Figure 16. Upon initialization, $\mathcal{B}$ internally generates a keypair $(\mathsf{vk},\mathsf{sk})$ which it sends to $\mathcal{A}$. When $\mathcal{A}$ queries the anamorphic-sign-then-anamorphic-decrypt oracle $O^{\mathcal{A}}_{\mathsf{aSign,aDec}}$ with $(\mathsf{msg},\mathsf{amsg})$, the reduction $\mathcal{B}$ queries its encrypt-then-decrypt oracle $O^{\mathcal{B}}_{\mathsf{Enc,Dec}}$ with amsg as the message and msg as the associated data. It receives a ciphertext ct and then computes $\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk},\mathsf{msg};\mathsf{ct})$, which it then sends to $\mathcal{A}$.

Observe that if $\mathcal{A}$ wins its anamorphic signature correctness game, then by the definition of RRep's anamorphic signing and decryption procedures it must have queried some $(\mathsf{msg},\mathsf{amsg})$ such that

$$\mathsf{prE.Dec}(\mathsf{k},\mathsf{msg},\mathsf{S.RRecov}(\mathsf{vk},\mathsf{sk},\mathsf{msg},\mathsf{S.Sign}(\mathsf{sk},\mathsf{msg};\mathsf{prE.Enc}(\mathsf{k},\mathsf{msg},\mathsf{amsg})))) \neq \mathsf{amsg}.$$

Per Definition 1, recovery always correctly extracts signing randomness (when vk, sk, and msg are fixed), hence it follows that $\mathsf{prE.Dec}(\mathsf{k},\mathsf{msg},\mathsf{prE.Enc}(\mathsf{k},\mathsf{msg},\mathsf{amsg})) \neq \mathsf{amsg}$ which means that $\mathcal{B}$ also wins its encryption correctness game. Clearly $\mathcal{B}$ is PPT since $\mathcal{A}$, $\mathsf{S.KeyGen}$, and $\mathsf{S.Sign}$ are PPT. The theorem bound follows from these observations. □

Theorem 13 ($\mathsf{RRep}^\star$ is $\mathsf{RoA\text{-}CAMA}$). *Let* S *be a randomness-recovering signature scheme and* prE *be a pseudorandom encryption scheme. Then* $\mathsf{aS} = \mathsf{RRep}^\star[\mathsf{S}, \mathsf{prE}]$ *(Construction 1) is stealthy. In particular, for all PPT adversaries* $\mathcal{A}$*, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{prE},\mathcal{B}}(\lambda).$$

Proof. At a high level, the theorem follows from the fact that $\mathsf{RRep}^\star$ replace the signing randomness with pseudorandom encryption scheme ciphertexts which appears uniformly random to an adversary that does not know the symmetric key, which is the $\mathsf{RRep}^\star$ double key. This argument is codified by considering a PPT reduction $\mathcal{B}$ against the $\mathsf{IND\$\text{-}CPA}$ security of prE constructed from a PPT distinguishing adversary $\mathcal{A}$ against the $\mathsf{RoA\text{-}CAMA}$ security of aS.

We first review the games that $\mathcal{A}$ and $\mathcal{B}$ play. The distinguisher $\mathcal{A}$ plays the $\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}$ game where it receives the signing keypair $(\mathsf{vk}, \mathsf{sk})$ (but crucially not the double key dk or any states, i.e., $\mathsf{st}_{\mathsf{aSign}}, \mathsf{st}_{\mathsf{aDec}}$), and has access to a real-or-anamorphic-signing oracle $O^{\mathcal{A}}_{\mathsf{Sign/aSign}}$ to which it queries messages and anamorphic messages and receive either real or anamorphic signatures. It must guess which it receives. The reduction $\mathcal{B}$ plays the $\mathcal{G}^{\mathsf{IND\$\text{-}CPA}}$ game where it accesses an encryption oracle $O^{\mathcal{B}}_{\mathsf{Enc}}$ to which it queries messages and obtain real or random ciphertexts, but it does not receive the symmetric key k or any states (i.e., $\mathsf{st}_{\mathsf{Enc}}$ or $\mathsf{st}_{\mathsf{Dec}}$). It aims to guess which kind of ciphertexts it receives.

We construct $\mathcal{B}$ from $\mathcal{A}$ as shown in Figure 16. Upon initialization, $\mathcal{B}$ internally generates a keypair $(\mathsf{vk}, \mathsf{sk})$ which it sends to $\mathcal{A}$. When $\mathcal{A}$ queries the real-or-anamorphic signing oracle $O^{\mathcal{A}}_{\mathsf{Sign/aSign}}$ with $(\mathsf{msg}, \mathsf{amsg})$, the reduction $\mathcal{B}$ queries its encryption oracle $O^{\mathcal{B}}_{\mathsf{Enc}}$ with message amsg and associated data msg and receives a ciphertext ct. It then uses the signing key to generate $\mathsf{sig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; \mathsf{ct})$. When $\mathcal{A}$ outputs a bit b^*, the reduction $\mathcal{B}$ returns $\neg b^*$ as its own output.

Observe that when $\mathcal{B}$ plays the $\mathcal{G}^{\mathsf{IND\$\text{-}CPA}}$ game with $b = 0$, it perfectly simulates the $\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}$ game with $b = 1$ to $\mathcal{A}$ because RRep generates k and $(\mathsf{st}_{\mathsf{Enc}}, \mathsf{st}_{\mathsf{Dec}})$ separately from the signing keypair and anamorphic signatures are signatures seeded by pseudorandom encryptions. When $\mathcal{B}$ plays the $\mathcal{G}^{\mathsf{IND\$\text{-}CPA}}$ game with $b = 1$, it receives random samples which it uses to seed signatures, so it perfectly simulates the $\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}$ game with $b = 0$ to $\mathcal{A}$ by providing it with fresh signatures. Clearly $\mathcal{B}$ is PPT since $\mathcal{A}$, $\mathsf{S.KeyGen}$, and $\mathsf{S.Sign}$ are PPT. The theorem bound follows from these observations. □

B.2 Details on RIdP, RIdP*, and RIdPX

Details on RIdP. We now analyze the correctness, stealthiness, and robustness (Definition 11) of RIdP, which was originally proposed as a construction for anamorphic encryption and which we have adapted to the anamorphic signature scheme setting. Beginning with correctness, we note that BGHMR [BGH+24] claim that RIdP (which they denote Σ_1) satisfies perfect correctness, however this is incorrect as the pseudorandom function may generate the same randomness r for distinct amsg and amsg' so that, while amsg is used to anamorphically sign, amsg' is the anamorphic message recovered. We capture the remedied correctness result for signature schemes in the following theorem.

Theorem 14 (RIdP is Correct). *Let* S *be a randomness-identifying signature scheme and* prF *be a pseudorandom function. Then* $\mathsf{aS} = \mathsf{RIdP}[\mathsf{S}, \mathsf{prF}]$ *(Construction 2) is correct. In particular, for all PPT adversaries* $\mathcal{A}$ *that make* $q_S \leq |\mathsf{aS}.\mathbb{ST}^{\mathsf{aSign}}_{\mathsf{pp}}|$ *anamorphic-sign-then-anamorphic-decrypt queries, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{B}}(\lambda) + q_S \frac{|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}| - 1}{|\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}|}.$$

Proof. We first recall that RIdP replaces signing randomness with pseudorandom function evaluations, i.e., $r \leftarrow \mathsf{prF}(\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{amsg})$, and decryption iterates through $r' \leftarrow \mathsf{prF}(\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{amsg}')$ for $\mathsf{amsg}' \in \mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}$ and returns the first amsg' that produces r' such that $r' = r$. Crucially, because dk is not known to the adversary in the correctness game $\mathcal{G}^{\mathsf{CORR}}$, then these r' appear random and the correctness of RIdP follows from the probability all r and r' generated in the game are distinct (because, per Definition 2, randomness identification of S requires that no distinct randomness can produce the same signature).[4] At

[4] We could alternatively have defined our correctness game such that the adversary knows dk and proven correctness of RIdP from a collision resistance property of the underlying pseudorandom function.

```
$\mathcal{G}_n$ for $n \in \{0,1,2\}$
$\mathsf{win} \leftarrow 0$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
$(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
$\mathsf{k} \leftarrow \mathsf{prF.KeyGen}(\mathsf{pp})$
$\mathsf{ctr}_{\mathsf{aSign}} \leftarrow 1$
$\mathsf{ctr}_{\mathsf{aDec}} \leftarrow 1$
$\mathbf{run}\ \mathcal{A}^{O^{\mathcal{A}}_{\mathsf{aSign},\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
$\mathbf{return}\ \mathsf{win}$
```

```
$O^{\mathcal{A}}_{\mathsf{aSign},\mathsf{aDec}}(\mathsf{msg}, \mathsf{amsg})$   $/\!\!/\ \mathcal{G}_{[3,\infty)}$
$r \leftarrow\!\$\ \mathsf{prF}.\mathbb{R}_{\mathsf{pp}}$
$\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$
$\mathbf{for}\ \mathsf{amsg}' \in \mathsf{aS}.\mathbb{AM}_{\mathsf{pp}} \setminus \{\mathsf{amsg}\}\ \mathbf{do}$
    $r' \leftarrow\!\$\ \mathsf{prF}.\mathbb{R}_{\mathsf{pp}}$
    $\mathbf{if}\ r' = r\ \mathbf{then}$
        $\mathsf{win} \leftarrow 1$
$\mathbf{return}\ \mathsf{asig}$
```

```
$O^{\mathcal{A}}_{\mathsf{aSign},\mathsf{aDec}}(\mathsf{msg}, \mathsf{amsg})$   $/\!\!/\ \mathcal{G}_{[0,3)}$
$r \leftarrow \mathsf{prF}(\mathsf{k}, (\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{amsg}))$   $/\!\!/\ \mathcal{G}_{[0,2)}$
$\mathsf{T}[(\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{amsg})] \leftarrow\!\$\ \mathsf{prF}.\mathbb{R}_{\mathsf{pp}}$   $/\!\!/\ \mathcal{G}_{[2,3)}$
$r \leftarrow \mathsf{T}[(\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{amsg})]$   $/\!\!/\ \mathcal{G}_{[2,3)}$
$\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$
$\mathsf{ctr}_{\mathsf{aSign}} \leftarrow \mathsf{ctr}_{\mathsf{aSign}} + 1$
$\mathsf{amsg}^* \leftarrow \bot$
$\mathbf{for}\ \mathsf{amsg}' \in \mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}\ \mathbf{do}$
    $r' \leftarrow \mathsf{prF}(\mathsf{k}, (\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{amsg}'))$
    $\mathbf{if}\ \mathsf{T}[(\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{amsg}')] = \bot\ \mathbf{then}$   $/\!\!/\ \mathcal{G}_{[2,3)}$
        $\mathsf{T}[(\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{amsg}')] \leftarrow\!\$\ \mathsf{prF}.\mathbb{R}_{\mathsf{pp}}$   $/\!\!/\ \mathcal{G}_{[2,3)}$
    $r' \leftarrow \mathsf{T}[(\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{amsg}')]$   $/\!\!/\ \mathcal{G}_{[2,3)}$
    $\mathbf{if}\ \mathsf{S.RIdtfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{asig}, r') = 1\ \mathbf{then}$   $/\!\!/\ \mathcal{G}_{[0,1)}$
    $\mathbf{if}\ r' = r\ \mathbf{then}$   $/\!\!/\ \mathcal{G}_{[1,3)}$
        $\mathsf{ctr}_{\mathsf{aDec}} \leftarrow \mathsf{ctr}_{\mathsf{aDec}} + 1$
        $\mathsf{amsg}^* \leftarrow \mathsf{amsg}'$
        $\mathbf{break}$
$\mathbf{if}\ \mathsf{amsg} \neq \mathsf{amsg}^*\ \mathbf{then}\ \mathsf{win} \leftarrow 1$
$\mathbf{return}\ \mathsf{asig}$
```

```
$\mathcal{B}^{O^{\mathcal{B}}_{\mathsf{prF}}}(\mathsf{pp})$
$\mathsf{win} \leftarrow 0$
$(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
$\mathsf{ctr}_{\mathsf{aSign}} \leftarrow 1$
$\mathsf{ctr}_{\mathsf{aDec}} \leftarrow 1$
$\mathbf{run}\ \mathcal{A}^{O^{\mathcal{A}}_{\mathsf{aSign},\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
$\mathbf{return}\ \mathsf{win}$
```

```
$O^{\mathcal{A}}_{\mathsf{aSign},\mathsf{aDec}}(\mathsf{msg}, \mathsf{amsg})$
$r \leftarrow O^{\mathcal{B}}_{\mathsf{PRF}}((\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{amsg}))$
$\mathsf{asig} \leftarrow \mathsf{aS.Sign}(\mathsf{sk}, \mathsf{msg}; r)$
$\mathsf{ctr}_{\mathsf{aSign}} \leftarrow \mathsf{ctr}_{\mathsf{aSign}} + 1$
$\mathsf{amsg}^* \leftarrow \bot$
$\mathbf{for}\ \mathsf{amsg}' \in \mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}\ \mathbf{do}$
    $r' \leftarrow O^{\mathcal{B}}_{\mathsf{PRF}}((\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{amsg}'))$
    $\mathbf{if}\ r' = r\ \mathbf{then}$
        $\mathsf{ctr}_{\mathsf{aDec}} \leftarrow \mathsf{ctr}_{\mathsf{aDec}} + 1$
        $\mathsf{amsg}^* \leftarrow \mathsf{amsg}'$
        $\mathbf{break}$
$\mathbf{if}\ \mathsf{amsg} \neq \mathsf{amsg}^*\ \mathbf{then}\ \mathsf{win} \leftarrow 1$
$\mathbf{return}\ \mathsf{asig}$
```

Fig. 17. Games (top) and reduction (bottom) used in the proof of Theorem 14

a high level, proof captures this intuition by performing a series hops beginning from a game $\mathcal{G}_0$ equivalent the correctness game $\mathcal{G}^{\mathsf{CORR}}$ on aS to a game $\mathcal{G}_3$ which no adversary can reliably win.

Game $\mathcal{G}_0$, shown in Figure 17, is simply the correctness game $\mathcal{G}^{\mathsf{CORR}}$ with the code for $\mathsf{aS} = \mathsf{RIdP}[\mathsf{S}, \mathsf{prF}]$ plugged in. The transition from game $\mathcal{G}_0$ to game $\mathcal{G}_1$ swaps from using $\mathsf{S.RIdtfy}$ to check whether r' was used to generate $\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$ to directly checking whether $r = r'$. These two games are equivalent per randomness identification (Definition 2), hence

$$\Pr[\mathcal{G}^{\mathsf{CORR}}_{\mathsf{aS},\mathcal{A}}(\lambda) \Rightarrow 1] = \Pr[\mathcal{G}_0(\lambda) \Rightarrow 1] = \Pr[\mathcal{G}_1(\lambda) \Rightarrow 1].$$

This transition from game $\mathcal{G}_1$ to game $\mathcal{G}_2$ swaps from generating each r in anamorphic signing and each r' in anamorphic decryption using prF to generating them using a lazily sampled true random function. The difference in advantage a PPT adversary $\mathcal{A}$ has between the games is bounded by the PRF advantage of a PPT adversary $\mathcal{B}$, which we construct as shown in Figure 17. Upon initialization, $\mathcal{B}$ generates a keypair $(\mathsf{vk}, \mathsf{sk})$ which it uses as input to run $\mathcal{A}$. It also initializes the counters $\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{ctr}_{\mathsf{aDec}}$ both to 0 and an internal variable win to false. When $\mathcal{A}$ queries the signing oracle $O^{\mathcal{A}}_{\mathsf{aSign},\mathsf{aDec}}$ with $(\mathsf{msg}, \mathsf{amsg})$, the reduction $\mathcal{B}$ queries $O^{\mathcal{B}}_{\mathsf{prF}}$ with $(\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{amsg})$ and receives a seed r. It then uses the signing key to generate $\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$ and increments $\mathsf{ctr}_{\mathsf{aSign}}$. The reduction $\mathcal{B}$ then sets $\mathsf{amsg}^* \leftarrow \perp$ and iteratively queries $O^{\mathcal{B}}_{\mathsf{prF}}$ with $(\mathsf{ctr}, \mathsf{amsg}')$ for $\mathsf{amsg}' \in \mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}$ and receives corresponding r', checking whether $r' = r$ for any amsg' and assigning amsg^* appropriately. Finally, the reduction checks whether $\mathsf{amsg} \neq \mathsf{amsg}^*$, sets $\mathsf{win} \leftarrow \mathsf{true}$ if this occurs, and also returns asig to the adversary. When $\mathcal{A}$ terminates, $\mathcal{B}$ outputs 0 if $\mathcal{A}$ wins and 1 otherwise. It is easy to see that when $\mathcal{B}$ plays the $\mathcal{G}^{\mathsf{PRF}}$ game with $b = 0$, it perfectly simulates game $\mathcal{G}_1$ to $\mathcal{A}$, while when $\mathcal{B}$ plays the $\mathcal{G}^{\mathsf{PRF}}$ game with $b = 1$, it perfectly simulates game $\mathcal{G}_2$ to $\mathcal{A}$. Furthermore, $\mathcal{B}$ is PPT since $\mathcal{A}$, $\mathsf{S.KeyGen}$, and $\mathsf{S.Sign}$ are PPT. It follows that

$$\Pr[\mathcal{G}_1(\lambda) \Rightarrow 1] - \Pr[\mathcal{G}_2(\lambda) \Rightarrow 1] = \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{B}}(\lambda).$$

The final transition from game $\mathcal{G}_2$ to game $\mathcal{G}_3$ performs code cleanup that does not change the functionality of the game. First note that $\mathsf{ctr}_{\mathsf{aDec}}$ always increments, because $\mathsf{amsg}' = \mathsf{amsg}$ will always trigger successful decryption if some other $\mathsf{amsg}' \neq \mathsf{amsg}$ does not. Furthermore, observe that the winning event only occurs when $\mathsf{amsg} \neq \mathsf{amsg}^*$, thus the for loop only needs to check $r' = r$ for r' generated by $\mathsf{amsg}' \neq \mathsf{amsg}$. With these changes, the random table is now only ever accessed on distinct inputs, as each anamorphic-sign-then-anamorphic-decrypt query accesses the table on a fixed $\mathsf{ctr}_{\mathsf{aSign}} = \mathsf{ctr}_{\mathsf{aDec}}$ and each anamorphic message in $\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}$ exactly once, each subsequent query uses a new $\mathsf{ctr}_{\mathsf{aSign}} = \mathsf{ctr}_{\mathsf{aDec}}$ value, and fewer than $|\mathsf{aS}.\mathbb{ST}^{\mathsf{aSign}}_{\mathsf{pp}}|$ queries are made so the counters never repeat. It follows that all randomnesses can be freshly sampled. Finally, we can move the winning condition check into the loop, thus removing the need to generate amsg^* at all. The probability $\mathcal{A}$ wins game $\mathcal{G}_3$ is simply the probability any sampled r and r' collide for any of the q_S anamorphic-sign-then-anamorphic-decrypt queries. Because $|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}| - 1$ different r' are sampled per query from a randomness set of size $|\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}|$ (since RIdP requires that $\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}} = \mathsf{prF}.\mathbb{R}_{\mathsf{pp}}$), it follows that

$$\Pr[\mathcal{G}_2(\lambda) \Rightarrow 1] \leq \Pr[\mathcal{G}_3(\lambda) \Rightarrow 1] \leq q_S \frac{|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}| - 1}{|\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}|}.$$

The theorem bound follows from these observations. □

We now analyze the stealthiness of RIdP. This proof is essentially identical to the proof of Lemma 4.1 from BGHMR, except adapted to signature schemes.

Theorem 15 (RIdP is $\mathsf{RoA\text{-}CAMA}$). *Let S be a randomness-identifying signature scheme and prF be a pseudorandom function. Then $\mathsf{aS} = \mathsf{RIdP}[\mathsf{S}, \mathsf{prF}]$ (Construction 2) is stealthy. In particular, for all PPT adversaries $\mathcal{A}$ that make $q_S \leq |\mathsf{aS}.\mathbb{ST}^{\mathsf{aSign}}_{\mathsf{pp}}|$ signing queries, there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{B}}(\lambda).$$

Proof. At a high level, stealthiness follows from the fact that RIdP replaces signing randomness with pseudorandom function outputs, which appear truly random to an adversary that does know the symmetric key, which RIdP uses as the double key. This is formalized by constructing a PPT distinguisher $\mathcal{B}$ against the PRF game from a PPT distinguisher $\mathcal{A}$ against the $\mathsf{RoA\text{-}CAMA}$ security of aS.

We construct $\mathcal{B}$ from $\mathcal{A}$ as shown in Figure 18. Upon initialization, $\mathcal{B}$ generates a keypair $(\mathsf{vk}, \mathsf{sk})$ which it uses as input to run $\mathcal{A}$. It also initializes a counter $\mathsf{ctr}_{\mathsf{aSign}} \leftarrow 1$. When $\mathcal{A}$ queries the signing oracle $O^{\mathcal{A}}_{\mathsf{Sign}}$ with $(\mathsf{msg}, \mathsf{amsg})$, the reduction $\mathcal{B}$ queries $O^{\mathcal{B}}_{\mathsf{prF}}$ with $(\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{amsg})$ and receives a seed r. It then uses the signing key to generate $\mathsf{sig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$ which it sends to $\mathcal{A}$, and it also updates $\mathsf{ctr}_{\mathsf{aSign}} \leftarrow \mathsf{ctr}_{\mathsf{aSign}} + 1$. When $\mathcal{A}$ outputs a bit b^*, $\mathcal{B}$ returns $\neg b^*$ as its own output.

```
$\mathcal{B}^{O^{\mathcal{B}}_{\mathsf{prF}}}(\mathsf{pp})$
    $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
    $\mathsf{ctr}_{\mathsf{aSign}} \leftarrow 1$
    $b^* \leftarrow \mathcal{A}^{O^{\mathcal{A}}_{\mathsf{Sign/aSign}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
    $\mathbf{return}\ \neg b^*$

$O^{\mathcal{A}}_{\mathsf{Sign/aSign}}(\mathsf{msg}, \mathsf{amsg})$
    $r \leftarrow \boxed{O^{\mathcal{B}}_{\mathsf{prF}}(\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{amsg})}$
    $\boxed{O^{\mathcal{B}}_{\mathsf{prF}}(\mathsf{ctr}_{\mathsf{aSign}}) \oplus \mathsf{amsg}}$
    $\mathsf{asig} \leftarrow \mathsf{aS.Sign}(\mathsf{sk}, \mathsf{msg}; r)$
    $\mathsf{ctr}_{\mathsf{aSign}} \leftarrow \mathsf{ctr}_{\mathsf{aSign}} + 1$
    $\mathbf{return}\ \mathsf{asig}$
```

Fig. 18. Reductions used in the proofs of Theorem 15 ($\boxed{\text{boxed}}$) and Theorem 20 ($\boxed{\text{dash boxed}}$)

Observe that when $\mathcal{B}$ plays the $\mathcal{G}^{\mathsf{PRF}}$ game with $b = 0$, it perfectly simulates the $\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}$ game with $b = 1$ to $\mathcal{A}$ because the RIdP transform generates k separately from the signing keypair and anamorphic signatures are simply signatures seeded by the pseudorandom function prF. When $\mathcal{B}$ plays the $\mathcal{G}^{\mathsf{PRF}}$ game with $b = 1$, it receives fresh random samples since $\mathsf{ctr}_{\mathsf{aSign}}$ is always updated (and $\mathcal{A}$ never makes more queries than the upper bound of $\mathsf{ctr}_{\mathsf{aSign}}$) hence each call to $O^{\mathcal{B}}_{\mathsf{prF}}$ is unique so $\mathcal{B}$ simulates the $\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}$ game with $b = 0$ to $\mathcal{A}$. Clearly $\mathcal{B}$ is PPT since $\mathcal{A}$, S.KeyGen, and S.Sign are PPT. The theorem bound follows from these observations. □

We now show that RIdP is robust. Note that BGHMR, in their proof of robustness for RIdP (Lemma 4.2), swap the prF usage to true random samples when generating r', incurring an additive PRF term in the advantage bound. This hop is unnecessary since robustness attempts to decrypt honest signatures, which already use fresh randomness.

Theorem 16 (RIdP is ROB-CMA). *Let* S *be a randomness-identifying signature scheme and* prF *be a pseudorandom function. Then* aS = RIdP[S, prF] *(Construction 2) is robust. In particular, for all PPT adversaries $\mathcal{A}$ that make at most q_S sign-then-anamorphic-decryption queries, there exists a PPT adversary such that*

$$\mathbf{Adv}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq q_S \frac{|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|}{|\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}|}.$$

Proof. We first review the robustness game. An adversary $\mathcal{A}$ playing the $\mathcal{G}^{\mathsf{ROB\text{-}CMA}}$ game receives nothing (aside from the public parameters) and has access to an oracle $O^{\mathcal{A}}_{\mathsf{Sign,aDec}}$ to which it queries messages and anamorphic decryption states and receives (attempted) anamorphic decryptions of honest signatures.

Now consider the probability that a PPT adversary $\mathcal{A}$ wins the $\mathcal{G}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aS}}$ game. Recall that $O^{\mathcal{A}}_{\mathsf{Sign,aDec}}$ generates an honest signature sig seeded by some fresh randomness r, and it then generates $|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|$ different asig' for seeds r' and returns the first amsg' such that $\mathsf{sig} = \mathsf{asig}'$, which is equivalent to $r = r'$ since S is randomness identifying (Definition 2). Observe that the distribution of r' does not matter because the r generated by the game is fresh. In particular, the probability, over the q_S queries, that the fresh randomness r used to generate the sig collides with any of the $|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|$ different r' used to generate a sig' (thus triggering an anamorphic decryption) is bounded by $q_S|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|/|\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}|$. □

Details on RIdP*. Theorems capturing the correctness and stealthiness of the RIdP* transform we propose are given below. Because the proofs are nearly identical to the proofs of Theorems 14 and 15, we simply highlight the changes needed from the corresponding proofs. We do not show robustness because Theorem 4 already establishes that RIdP* achieve dictator unforgeability.

Theorem 17 (RIdP* is Correct). *Let* S *be a randomness-identifying signature scheme and* prF *be a pseudorandom function. Then* $\mathsf{aS} = \mathsf{RIdP}^\star[\mathsf{S}, \mathsf{prF}]$ *(Construction 2) is correct. In particular, for all PPT adversaries* $\mathcal{A}$ *that make* $q_S \leq |\mathsf{aS}.\mathbb{ST}_{\mathsf{pp}}^{\mathsf{aSign}}|$ *anamorphic-sign-then-anamorphic-decrypt queries, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}_{\mathsf{aS},\mathcal{A}}^{\mathsf{CORR}}(\lambda) = \mathbf{Adv}_{\mathsf{prF},\mathcal{B}}^{\mathsf{PRF}}(\lambda) + q_S \frac{|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}| - 1}{|\mathsf{S}.\mathbb{R}_{\mathsf{pp}}^{\mathsf{Sign}}|}.$$

Proof. The proof follows essentially identically to the proof of Theorem 14 which shows that RIdP is correct. In that proof, a series of game hops is performed where the first applies the definition of randomness identification, the second swaps from using prF to using lazy samples, and the final performs code cleanup. Because RIdP* differs from RIdP by including msg as input to the prF and, within each run of the decryption loop msg is fixed, analogous transitions work for RIdP* as well. □

Theorem 18 (RIdP* is RoA-CAMA). *Let* S *be a randomness-identifying signature scheme and* prF *be a pseudorandom function. Then* $\mathsf{aS} = \mathsf{RIdP}^\star[\mathsf{S}, \mathsf{prF}]$ *(Construction 2) is stealthy. In particular, for all PPT adversaries* $\mathcal{A}$ *that make* $q_S \leq |\mathsf{aS}.\mathbb{ST}_{\mathsf{pp}}^{\mathsf{aSign}}|$ *signing queries, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}_{\mathsf{aS},\mathcal{A}}^{\mathsf{RoA\text{-}CAMA}}(\lambda) \leq \mathbf{Adv}_{\mathsf{prF},\mathcal{B}}^{\mathsf{PRF}}(\lambda).$$

Proof. The proof follows essentially identically to the proof of Theorem 15 which shows that RIdP is stealthy. In that proof, each signature is seeded with a prF output which takes in a counter that increments each query. Because RIdP* only differs from RIdP by adding additional inputs to the prF, the counter is unaffected and the reduction for that proof works for RIdP* as well. □

Details on RIdPX. We now analyze the correctness, stealthiness, and robustness of RIdPX, which again was originally proposed as a construction for anamorphic encryption and which we have adapted to the anamorphic signature scheme setting.

Theorem 19 (RIdPX is Correct). *Let* S *be a randomness-identifying signature scheme and* prF *be a pseudorandom function. Then* $\mathsf{aS} = \mathsf{RIdPX}[\mathsf{S}, \mathsf{prF}]$ *(Construction 2) is correct. In particular, for all PPT adversaries* $\mathcal{A}$

$$\mathbf{Adv}_{\mathsf{aS},\mathcal{A}}^{\mathsf{CORR}}(\lambda) = 0.$$

Proof. We can analyze the correctness game on aS directly. As before, because S is randomness identifying (Definition 2), distinct randomness cannot seed the same signature. We thus need only consider the probability randomness itself collides. Recall that RIdPX anamorphically signs by seeding an honest signature with randomness $r \leftarrow \mathsf{prF}(\mathsf{ctr}_{\mathsf{aSign}}) \oplus \mathsf{amsg}$ and anamorphically decrypts by finding the amsg' such that $r = \mathsf{prF}(\mathsf{ctr}_{\mathsf{aDec}}) \oplus \mathsf{amsg}'$. Because the correctness game proceeds by encrypting once then decrypting once, and decryption always succeeds (r corresponding to amsg will decrypt if no r' corresponding to $\mathsf{amsg}' \neq \mathsf{amsg}$ does), then $\mathsf{ctr}_{\mathsf{aSign}} = \mathsf{ctr}_{\mathsf{aDec}}$ always. It follows that $\mathsf{prF}(\mathsf{ctr}_{\mathsf{aSign}}) = \mathsf{prF}(\mathsf{ctr}_{\mathsf{aDec}})$ for each query. Let r^* denote this quantity. Then, since $\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}$ is a group there cannot exist amsg and amsg' such that $r^* \oplus \mathsf{amsg} = r^* \oplus \mathsf{amsg}'$. The theorem bound follows from these observations. □

Theorem 20 (RIdPX is RoA-CAMA). *Let* S *be a randomness-identifying signature scheme and* prF *be a pseudorandom function. Then* $\mathsf{aS} = \mathsf{RIdPX}[\mathsf{S}, \mathsf{prF}]$ *(Construction 2) is stealthy. In particular, for all PPT adversaries* $\mathcal{A}$ *that make* $q_S \leq |\mathsf{aS}.\mathbb{ST}_{\mathsf{pp}}^{\mathsf{aSign}}|$ *signing queries, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}_{\mathsf{aS},\mathcal{A}}^{\mathsf{RoA\text{-}CAMA}}(\lambda) \leq \mathbf{Adv}_{\mathsf{prF},\mathcal{B}}^{\mathsf{PRF}}(\lambda).$$

Proof. The proof follows similarly to the proof of Theorem 15. In particular, we construct $\mathcal{B}$ from $\mathcal{A}$ as shown in Figure 18. Upon initialization, $\mathcal{B}$ generates a keypair $(\mathsf{vk}, \mathsf{sk})$ which it sends to $\mathcal{A}$ and initializes a counter $\mathsf{ctr}_{\mathsf{aSign}} \leftarrow 1$. When $\mathcal{A}$ queries $O_{\mathsf{Sign}}^{\mathcal{A}}$ with $(\mathsf{msg}, \mathsf{amsg})$, the reduction $\mathcal{B}$ queries $O_{\mathsf{prF}}^{\mathcal{B}}$ with $\mathsf{ctr}_{\mathsf{aSign}}$ and receives a seed (say r''). It then computes $r \leftarrow r'' \oplus \mathsf{amsg}$ and generates $\mathsf{sig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$ which it sends to $\mathcal{A}$. It also updates $\mathsf{ctr}_{\mathsf{aSign}} \leftarrow \mathsf{ctr}_{\mathsf{aSign}} + 1$. When $\mathcal{A}$ outputs a bit b^*, the reduction $\mathcal{B}$ returns $\neg b^*$ as its own output.

Observe that when $\mathcal{B}$ plays the $\mathcal{G}^{\mathsf{PRF}}$ game with $b = 0$, it simply runs the RIdP code and hence simulates $\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}$ with $b = 1$ to $\mathcal{A}$. When $\mathcal{B}$ plays the $\mathcal{G}^{\mathsf{PRF}}$ game with $b = 1$, it receives fresh random r' which masks amsg to produce fresh random r. This r is then used to seed signatures, i.e., it performs the real signing procedure, so it simulates the $\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}$ game with $b = 0$ to $\mathcal{A}$. Clearly $\mathcal{B}$ is PPT since $\mathcal{A}$, $\mathsf{S.KeyGen}$, and $\mathsf{S.Sign}$ are PPT. The theorem bound follows from these observations. □

Theorem 21 (RIdPX is ROB-CMA). *Let* S *be a randomness-identifying signature scheme and* prF *be a pseudorandom function. Then* $\mathsf{aS} = \mathsf{RIdPX}[\mathsf{S}, \mathsf{prF}]$ *(Construction 2) is robust. In particular, for all PPT adversaries* $\mathcal{A}$ *that make up to* q_S *sign-then-anamorphic-decryption queries, there exists a PPT adversary such that*

$$\mathbf{Adv}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq q_S \frac{|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|}{|\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}|}.$$

Proof. The proof follows similarly to the proof of Theorem 16. Consider the probability that a PPT adversary $\mathcal{A}$ wins game robustness game on $\mathsf{aS} = \mathsf{RIdPX}[\mathsf{S}, \mathsf{prF}]$. Recall that $O^{\mathcal{A}}_{\mathsf{Sign,aDec}}$ generates an honest signature sig seeded by some fresh randomness r, and it then generates $|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|$ different asig' for seeds $r' \leftarrow r'' \oplus \mathsf{amsg}'$ and returns amsg' if $\mathsf{sig} = \mathsf{asig}'$ (or equivalently $r = r'$, since S is randomness identifying) at any point. The distribution of r' does not matter because the r generated by the game is fresh. Hence, the probability, over the q_S queries, that the fresh randomness r used to generate the sig collides with any of the $|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|$ different r' used to generate a asig' (thus triggering an anamorphic decryption) is bounded by $q_S|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|/|\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}|$. □

As with RIdP, the corresponding robustness proof in [BGH+24] for the anamorphic encryption version of RIdPX (Lemma 4.4) incurs an extra additive PRF term in the advantage bound.

C Details on Strengthening Robustness to Dictator Unforgeability

C.1 Full Proof of Theorem 2 (IND-CAMA + DUF-CASA ⇒ IND-CASA)

In Section 4, we introduced a comprehensive dictator unforgeability notion ($\mathsf{DUF\text{-}CASA}$) and briefly discussed how it relates to CCA-style anamorphic message confidentiality ($\mathsf{IND\text{-}CASA}$). Here, we provide full proofs of this claim. We begin by introducing a useful intermediate multi-challenge dictator unforgeability notion and showing it is implied by single-challenge dictator unforgeability. Then we use this multi-challenge dictator unforgeability notion in the main proof.

Single- and Multi-Challenge Dictator Unforgeability. We now present a multi-challenge notion of dictator unforgeability and show that it is implied by (single-challenge) dictator unforgeability we have discussed thus far. In the multi-challenge game shown in Figure 19, the dictator wins if it makes any non-trivial anamorphic decryption query that does not output $\perp$. Similar to single-challenge dictator unforgeability, the multi-challenge definition is parameterized by a triviality predicate $\mathsf{pred}_{\mathsf{trivial}}$ that encapsulates what counts as a trivial forward. This $\mathsf{pred}_{\mathsf{trivial}}$ is evaluated every anamorphic decryption query. We define the $\mathsf{mc\text{-}DUF\text{-}CASA}$ advantage for an anamorphic signature scheme aS and triviality predicate $\mathsf{pred}_{\mathsf{trivial}}$ by

$$\mathbf{Adv}^{\mathsf{mc\text{-}DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \Rightarrow 1].$$

Theorem 22 (DUF-CASA ⇔ mc-DUF-CASA). *Let* aS *be an anamorphic signature scheme. It is dictator unforgeable for triviality predicate* $\mathsf{pred}_{\mathsf{trivial}}$ *if and only if it is also multi-challenge dictator unforgeable for* $\mathsf{pred}_{\mathsf{trivial}}$*. In particular, for all PPT multi-challenge dictator unforgeability adversaries* $\mathcal{A}$ *that make at most* q_D *anamorphic decryption queries, there exists a PPT dictator unforgeability adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{mc\text{-}DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \leq q_D \mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{B}}(\lambda).$$

```
$\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda)$
    $\mathsf{win} \leftarrow 0$
    $(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}}) \leftarrow ([\cdot], [\cdot])$
    $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
    $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aSign}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow \mathsf{aS.aKeyGen}(\mathsf{pp})$
    $\textbf{run } \mathcal{A}^{O_{\mathsf{aSign}}, O_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
    $\textbf{return } \mathsf{win}$

$O_{\mathsf{aSign}}(\mathsf{msg}, \mathsf{amsg})$
    $\mathsf{asig} \leftarrow \mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aSign}})$
    $\mathsf{L}_{\mathsf{aSign}}.\mathsf{add}((\mathsf{msg}, \mathsf{asig}))$
    $\textbf{return } \mathsf{asig}$

$O_{\mathsf{aDec}}(\mathsf{msg}, \mathsf{asig})$
    $\mathsf{amsg} \leftarrow \mathsf{aS.aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig} : \mathsf{st}_{\mathsf{aDec}})$
    $\mathsf{L}_{\mathsf{aDec}}.\mathsf{add}((\mathsf{msg}, \mathsf{asig}))$
    $b_{\mathsf{valid}} \leftarrow [\![\mathsf{amsg} \neq \bot]\!]$
    $b_{\mathsf{trivial}} \leftarrow \mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}})$
    $\textbf{if } b_{\mathsf{valid}} \wedge \neg b_{\mathsf{trivial}} \textbf{ then } \mathsf{win} \leftarrow 1$
    $\textbf{return } \mathsf{amsg}$
```

Fig. 19. Multi-challenge dictator unforgeability game $\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CASA}}$

Furthermore, for all PPT dictator unforgeability adversaries $\mathcal{B}$, there exists a PPT multi-challenge dictator unforgeability adversary $\mathcal{A}$ such that

$$\mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{B}}(\lambda) = \mathbf{Adv}^{\mathsf{mc\text{-}DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda).$$

Proof. The second part of the theorem follows immediately by defining $\mathcal{A}^{O^{\mathcal{A}}_{\mathsf{aSign}}, O^{\mathcal{A}}_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$ to run $(\mathsf{msg}^*, \mathsf{asig}^*) \leftarrow \mathcal{B}^{O^{\mathcal{A}}_{\mathsf{aSign}}, O^{\mathcal{A}}_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$ then query $O^{\mathcal{A}}_{\mathsf{aDec}}(\mathsf{msg}^*, \mathsf{asig}^*)$. Clearly, $\mathcal{A}$ gives $\mathcal{B}$ the correct view and wins when $\mathcal{B}$ would win.

To handle the first part of the theorem, let $\mathcal{A}$ be a PPT adversary against the $\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CASA}}$ game and assume, without loss of generality, that it always makes precisely q_D anamorphic decryption queries. The proof proceeds by viewing win as the combination of q_D bad events $\mathsf{win} = \mathsf{bad}_1 \vee \cdots \vee \mathsf{bad}_{q_D}$ where each bad_i corresponds to $\mathcal{A}$ providing a successful forgery in its i-th anamorphic decryption query. For each i we can construct a $\mathcal{B}_i$ from $\mathcal{A}$ against the $\mathsf{DUF\text{-}CASA}$ security of aS which wins whenever bad_i would be set. The overall bound follows by letting $\mathcal{B}$ be a PPT algorithm that randomly samples one of the $\mathcal{B}_i$ and uses it against the $\mathsf{DUF\text{-}CASA}$ security of aS.

We first review the games that $\mathcal{A}$ and $\mathcal{B}$ play. The adversary $\mathcal{A}$ plays the $\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CASA}}$ game where it receives the signing keypair $(\mathsf{vk}, \mathsf{sk})$ and has access to an anamorphic signing oracle $O^{\mathcal{A}}_{\mathsf{aSign}}$ to which it can query messages and anamorphic messages and receive anamorphic signatures. It also receives access to an anamorphic decryption oracle $O^{\mathcal{A}}_{\mathsf{aDec}}$ which it queries with $(\mathsf{msg}, \mathsf{asig})$ and wins if any such query is non-trivial (as determined by $\mathsf{pred}_{\mathsf{trivial}}$) and anamorphically decrypts to anything other than $\bot$. The reduction $\mathcal{B}$ plays the $\mathcal{G}^{\mathsf{DUF\text{-}CASA}}$ game where it has access to the same oracles $O^{\mathcal{B}}_{\mathsf{aSign}}$ and $O^{\mathcal{B}}_{\mathsf{aDec}}$ but returns exactly one forgery $(\mathsf{msg}^*, \mathsf{asig}^*)$ that it hopes will anamorphically decrypt successfully.

Letting bad_i denote the bad event that win is set in $\mathcal{A}$'s i-th oracle query, we have

$$\Pr[\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \Rightarrow 1] \leq \Pr[\mathsf{bad}_1 \vee \cdots \vee \mathsf{bad}_{q_D}] \leq \sum_{i=1}^{q_D} \Pr[\mathsf{bad}_i].$$

We can bound the probability of bad_n by the $\mathsf{DUF\text{-}CASA}$ advantage of the reduction $\mathcal{B}_n$ shown in Figure 20. It forwards all of the $\mathcal{A}$'s oracle queries until the n-th decryption query. Then it halts running $\mathcal{A}$ and outputs this query as its attempted forgery. Note that $\mathcal{B}_n$ perfectly simulates the view of $\mathcal{A}$ and $\mathcal{B}_n$ wins precisely when bad_n would be set so $\Pr[\mathsf{bad}_i] = \mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{B}_n}(\lambda)$. We have that

$$\Pr[\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \Rightarrow 1] \leq \sum_{n=1}^{q_D} \mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{B}_n}(\lambda).$$

```
$\mathcal{B}_n^{O_{\mathsf{aSign}}^{\mathcal{B}_n}, O_{\mathsf{aDec}}^{\mathcal{B}_n}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
    $i_D \leftarrow 0$
    $(\mathsf{msg}^*, \mathsf{asig}^*) \leftarrow (\perp, \perp)$
    run $\mathcal{A}^{O_{\mathsf{aSign}}^{\mathcal{A}}, O_{\mathsf{aDec}}^{\mathcal{A}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
    return $(\mathsf{msg}^*, \mathsf{asig}^*)$

$O_{\mathsf{aSign}}^{\mathcal{A}}(\mathsf{msg}, \mathsf{amsg})$
    $\mathsf{asig} \leftarrow O_{\mathsf{aSign}}^{\mathcal{B}_n}$
    return $\mathsf{asig}$

$O_{\mathsf{aDec}}^{\mathcal{A}}(\mathsf{msg}, \mathsf{asig})$
    $i_D \leftarrow i_D + 1$
    if $i_D = n$ then
        $(\mathsf{msg}^*, \mathsf{asig}^*) \leftarrow (\mathsf{msg}, \mathsf{asig})$
        halt $\mathcal{A}$
    $\mathsf{amsg} \leftarrow O_{\mathsf{aDec}}^{\mathcal{B}_n}(\mathsf{msg}, \mathsf{asig})$
    return $\mathsf{amsg}$
```

```
$\mathcal{B}^{O_{\mathsf{aSign}}^{\mathcal{B}}, O_{\mathsf{aDec}}^{\mathcal{B}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
    $n \leftarrow\$ \{1, \ldots, q_D\}$
    return $\mathcal{B}_n^{O_{\mathsf{aSign}}^{\mathcal{B}}, O_{\mathsf{aDec}}^{\mathcal{B}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
```

Fig. 20. Reductions used in the proof of Theorem 22

We now construct $\mathcal{B}$ as shown in Figure 20 which samples n and then runs $\mathcal{B}_n$ for its attack. Clearly $\mathcal{B}$ is PPT. Since each i^* has a $1/q_D$ chance of being selected it follows that

$$\mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS}, \mathsf{pred}_{\mathsf{trivial}}, \mathcal{B}}(\lambda) = \frac{1}{q_D} \sum_{i=1}^{q_D} \mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS}, \mathsf{pred}_{\mathsf{trivial}}, \mathcal{B}_i}(\lambda).$$

The theorem bound follows from these observations. □

Dictator Unforgeability and Anamorphic Message Confidentiality. We now show that dictator unforgeability and indistinguishability under chosen anamorphic message attack implies indistinguishability under chosen anamorphic signature attack. This effectively follows the ubiquitous result that ciphertext integrity (which inspired dictator unforgeability) and CPA security implies CCA security.

Theorem 2 (IND-CAMA + DUF-CASA ⇒ IND-CASA). *Let* aS *be an* IND-CAMA *indistinguishable and dictator unforgeable anamorphic signature scheme for triviality predicate* $\mathsf{pred}_{\mathsf{trivial}}$*. Then it is also* IND-CASA *indistinguishable for* $\mathsf{pred}_{\mathsf{trivial}}$*. In particular, for all PPT adversaries* $\mathcal{A}$ *that make at most* q_D *anamorphic decryption queries, there exist PPT adversaries* $\mathcal{B}_0$ *and* $\mathcal{B}_1$ *such that*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CASA}}_{\mathsf{aS}, \mathsf{pred}_{\mathsf{trivial}}, \mathcal{A}}(\lambda) \leq 2 q_D \, \mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS}, \mathsf{pred}_{\mathsf{trivial}}, \mathcal{B}_0}(\lambda) + \mathbf{Adv}^{\mathsf{IND\text{-}CAMA}}_{\mathsf{aS}, \mathcal{B}_1}(\lambda).$$

Proof. The proof proceeds by first hopping from a game $\mathcal{G}_0$, equivalent to the $\mathcal{G}^{\mathsf{IND\text{-}CASA}}$ game on aS, to a game $\mathcal{G}_1$ which responds to all anamorphic decryption queries with $\perp$. This transition is bounded by the mc-DUF-CASA advantage of a PPT reduction $\mathcal{B}_0'$ which itself is bounded by the DUF-CASA advantage of a PPT $\mathcal{B}_0$ per Theorem 22. Since the final game $\mathcal{G}_1$ never answers any anamorphic decryption queries, defeating this game is equivalent to breaking IND-CAMA security. In particular, given a PPT adversary $\mathcal{A}$ playing $\mathcal{G}_1$ we can construct a PPT $\mathcal{B}_1$ that simulates $\mathcal{G}_1$ to $\mathcal{A}$ and uses it to win the $\mathcal{G}^{\mathsf{IND\text{-}CAMA}}$ game.

We first review the games that $\mathcal{A}$, $\mathcal{B}_0'$, and $\mathcal{B}_1$ play. The (two-stage) adversary $\mathcal{A}$ plays the $\mathcal{G}^{\mathsf{IND\text{-}CASA}}$ game where it receives $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$ and outputs a message msg^* and two anamorphic messages amsg_0^* and amsg_1^*. The game samples a random bit b and gives an anamorphic signature asig^* containing amsg_b encrypted within it to $\mathcal{A}$ who must guess which anamorphic message is hidden inside. In both stages, $\mathcal{A}$ receives access to an anamorphic signing oracle $O_{\mathsf{aSign}}^{\mathcal{A}}$ and anamorphic decryption oracle $O_{\mathsf{aDec}}^{\mathcal{A}}$. The (two-stage) reduction $\mathcal{B}_1$ plays the $\mathcal{G}^{\mathsf{IND\text{-}CAMA}}$ game which is identical except that $\mathcal{B}_1$ does not receive access to the anamorphic decryption oracle. Finally, the reduction $\mathcal{B}_0'$ plays the $\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CASA}}$ game where it receives $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$ and access to an anamorphic signing oracle $O_{\mathsf{aSign}}^{\mathcal{B}_0'}$ and anamorphic decryption oracle $O_{\mathsf{aDec}}^{\mathcal{B}_0'}$ and it wins if any of its non-trivial queries to the anamorphic decryption oracle are successful (i.e., respond with an $\mathsf{amsg} \neq \perp$).

```
$\mathcal{G}_n$ for $n \in \{0,1\}$
$b \leftarrow\$ \{0,1\}$
$(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}}) \leftarrow ([\cdot],[\cdot])$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
$(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aSign}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow \mathsf{aS.aKeyGen}(\mathsf{pp})$
$(\mathsf{msg}^*, \mathsf{amsg}^*_0, \mathsf{amsg}^*_1, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_0^{O_{\mathsf{aSign}}, O_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
$\mathsf{asig}^* \leftarrow \mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}^*, \mathsf{amsg}^*_b : \mathsf{st}_{\mathsf{aSign}})$
$\mathsf{L}_{\mathsf{aSign}}.\mathsf{add}((\mathsf{msg}^*, \mathsf{asig}^*))$
$b^* \leftarrow \mathcal{A}_1^{O_{\mathsf{aSign}}, O_{\mathsf{aDec}}}(\mathsf{st}_{\mathcal{A}}, \mathsf{asig}^*)$
$\textbf{return } [\![ b = b^* ]\!]$

$O_{\mathsf{aSign}}(\mathsf{msg}, \mathsf{amsg})$
$\mathsf{asig} \leftarrow \mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aSign}})$
$\mathsf{L}_{\mathsf{aSign}}.\mathsf{add}((\mathsf{msg}, \mathsf{asig}))$
$\textbf{return } \mathsf{asig}$

$O_{\mathsf{aDec}}(\mathsf{msg}, \mathsf{asig})$
$\mathsf{amsg} \leftarrow \mathsf{aS.aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig} : \mathsf{st}_{\mathsf{aDec}})$
$\mathsf{L}_{\mathsf{aDec}}.\mathsf{add}((\mathsf{msg}, \mathsf{asig}))$
$\textbf{if } \mathsf{amsg} \neq \bot \wedge \mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}}) = 0 \textbf{ then}$
    $\mathsf{bad} \leftarrow \mathsf{true}$
    $\textbf{return } \mathsf{amsg}$    // $\mathcal{G}_0$
$\textbf{return } \bot$
```

```
$\mathcal{B}_0'^{\,O^{\mathcal{B}'_0}_{\mathsf{aSign}}, O^{\mathcal{B}'_0}_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
$(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}}) \leftarrow ([\cdot],[\cdot])$
$(\mathsf{msg}^*, \mathsf{amsg}^*_0, \mathsf{amsg}^*_1, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_0^{O^{\mathcal{B}'_0}_{\mathsf{aSign}}, O^{\mathcal{A}}_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
$b \leftarrow\$ \{0,1\}$
$\mathsf{asig}^* \leftarrow O^{\mathcal{B}'_0}_{\mathsf{aSign}}(\mathsf{msg}^*, \mathsf{amsg}^*_b)$
$\textbf{run } \mathcal{A}_1^{O^{\mathcal{B}'_0}_{\mathsf{aSign}}, O^{\mathcal{A}}_{\mathsf{aDec}}}(\mathsf{st}_{\mathcal{A}}, \mathsf{asig}^*)$

$O^{\mathcal{A}}_{\mathsf{aDec}}(\mathsf{msg}, \mathsf{asig})$
$O^{\mathcal{B}'_0}_{\mathsf{aDec}}(\mathsf{msg}, \mathsf{asig})\textbf{return } \bot$
```

```
$\mathcal{B}_{1,0}^{\,O^{\mathcal{B}_1}_{\mathsf{aSign}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
$\textbf{return } \mathcal{A}_0^{O^{\mathcal{B}_1}_{\mathsf{aSign}}, O^{\mathcal{A}}_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$

$\mathcal{B}_{1,1}^{\,O^{\mathcal{B}_1}_{\mathsf{aSign}}}(\mathsf{st}_{\mathcal{A}}, \mathsf{asig}^*)$
$\textbf{return } \mathcal{A}_1^{O^{\mathcal{B}_1}_{\mathsf{aSign}}, O^{\mathcal{A}}_{\mathsf{aDec}}}(\mathsf{st}_{\mathcal{A}}, \mathsf{asig}^*)$

$O^{\mathcal{A}}_{\mathsf{aDec}}(\mathsf{msg}, \mathsf{asig})$
$\textbf{return } \bot$
```

Fig. 21. Games (top) and reductions (bottom left, bottom right) used in the proof of Theorem 2

Now consider a PPT adversary $\mathcal{A}$ playing game $\mathcal{G}_0$ or game $\mathcal{G}_1$, shown in Figure 21. Game $\mathcal{G}_0$ is equivalent to $\mathcal{G}^{\mathsf{IND\text{-}CASA}}$. Games $\mathcal{G}_0$ and $\mathcal{G}_1$ are identical until the event bad occurs where $\mathcal{A}$ makes a non-trivial anamorphic decryption query that doesn't return $\bot$. We construct $\mathcal{B}'_0$, as shown in Figure 21, that wins if bad occurs as follows. In runs both stages of $\mathcal{A}$ using its own anamorphic signing oracle to respond to oracle queries. It forwards $\mathcal{A}$'s anamorphic decryption queries to its own oracle, then simply returns $\bot$. When $\mathcal{A}$ outputs $(\mathsf{msg}^*, \mathsf{amsg}^*_0, \mathsf{amsg}^*_1, \mathsf{st}_{\mathcal{A}})$, it randomly samples a bit b and sends $(\mathsf{msg}^*, \mathsf{amsg}^*_b)$ to $O^{\mathcal{B}'_0}_{\mathsf{aSign}}$ to obtains asig^*. Since all queries are forwarded (and the challenge is added to $\mathsf{L}_{\mathsf{aSign}}$ in the $\mathcal{G}^{\mathsf{IND\text{-}CASA}}$ game) then any non-trivial query $\mathcal{A}$ makes to its anamorphic decryption query that doesn't output $\bot$ is also non-trivial for $\mathcal{B}'_0$, causing it to win the $\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CASA}}$ game. Note that $\mathcal{B}'_0$ is PPT because it only ever forwards queries or outputs made by $\mathcal{A}$, which is PPT. Furthermore, it follows, after additionally applying Theorem 22, that there exists a PPT $\mathcal{B}_0$ such that

$$\Pr[\mathcal{G}_0(\lambda)] - \Pr[\mathcal{G}_1(\lambda)] \leq \Pr[\mathsf{bad}] = \mathbf{Adv}^{\mathsf{mc\text{-}DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{B}'_0}(\lambda) \leq q_D \mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{B}_0}(\lambda).$$

We now bound the advantage $\mathcal{A}$ can have against game $\mathcal{G}_1$ by the $\mathsf{IND\text{-}CAMA}$ advantage of a PPT reduction $\mathcal{B}_1$, which we construct as shown in Figure 21. It simply runs $\mathcal{A}$ forwarding anamorphic signing queries to its own oracle and responding to anamorphic decryption queries with $\bot$. Clearly $\mathcal{B}_1$ is PPT because it only ever forwards queries or outputs made by $\mathcal{A}$, which is PPT. Furthermore, since $\mathcal{B}_1$ simulates game $\mathcal{G}_1$ to $\mathcal{A}$ and plays the same challenges it has given to $\mathcal{A}$, it follows that

$$\Pr[\mathcal{G}_1(\lambda)] = \Pr[\mathcal{G}^{\mathsf{IND\text{-}CAMA}}_{\mathsf{aS},\mathcal{B}_1}] = \frac{1}{2}\mathbf{Adv}^{\mathsf{IND\text{-}CAMA}}_{\mathsf{aS},\mathcal{B}_1}(\lambda) + \frac{1}{2}.$$

The theorem bound follows from these observations. □

C.2 Full Proof of Theorem 3 ($\mathsf{RRep}^\star$ is $\mathsf{DUF\text{-}CASA}$)

In Section 4, we argued that the improved randomness replacement transform $\mathsf{RRep}^\star$ is dictator unforgeable. We provide a full proof here.

Theorem 3 ($\mathsf{RRep}^\star$ is $\mathsf{DUF\text{-}CASA}$). *Let* S *be a strongly randomness-recovering signature scheme and* prE *be an AEAD scheme achieving ciphertext integrity for equality-pattern-respecting triviality predicate* $\mathsf{pred}_{\mathsf{trivial}}$ *(which is permissive if* prE *is). Then* $\mathsf{aS} = \mathsf{RRep}^\star[\mathsf{S}, \mathsf{prE}]$ *(Construction 1) is dictator unforgeable for* $\mathsf{pred}_{\mathsf{trivial}}$*. In particular, for all PPT adversaries* $\mathcal{A}$*, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{INT\text{-}CT}}_{\mathsf{prE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{B}}(\lambda).$$

Proof. Let $\mathcal{A}$ be a PPT adversary and consider the sequence of games $\mathcal{G}_0$ through $\mathcal{G}_2$ in Figure 22. We claim that $\mathcal{G}_0$ is equivalent to the $\mathcal{G}^{\mathsf{DUF\text{-}CASA}}$ game on $\mathsf{RRep}^\star[\mathsf{S}, \mathsf{prE}]$. It was obtained by plugging in the code of $\mathsf{RRep}^\star$, then making explicit the behavior of encryption when recovering an anamorphic ciphertext from an anamorphic ciphertext provided by $\mathcal{A}$ fails rather than calling prE on a $\perp$ input. In particular, we return $\perp$ early in O_{aDec} and return 0 early at the end of the game, because we know that decryption will return $\perp$. In the case that prE is strict, we additionally set a flag rej that will cause us to similarly reject all future inputs. The highlighted sk indicates where it is not needed if S is publicly randomness recovering. We have that

$$\Pr[\mathcal{G}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{RRep}^\star[\mathsf{S},\mathsf{prE}],\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \Rightarrow 1] = \Pr[\mathcal{G}_0(\lambda) \Rightarrow 1].$$

Moving from game $\mathcal{G}_0$ to game $\mathcal{G}_1$, we stop adding $(\mathsf{msg}, \mathsf{asig})$ to $\mathsf{L}_{\mathsf{aSign}}$ when recovering a ciphertext from asig failed. For a strict scheme, rej will anyway be set in this case. This will anyway prevent $\mathsf{pred}_{\mathsf{trivial}}$ from being run at the end of the game. For a permissive scheme, this will not change the later output of $\mathsf{pred}_{\mathsf{trivial}}$ because it is permissive. Let $\mathsf{L}_{\mathsf{aSign}}$ denote the value of this list at the end of either $\mathcal{G}_0$ or $\mathcal{G}_1$ and let $\mathsf{L}^n_{\mathsf{aDec}}$ denote the value of this list at the end of $\mathcal{G}_n$. We claim $\mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}^0_{\mathsf{aDec}}) = \mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}^1_{\mathsf{aDec}})$ will hold whenever the predicate would be evaluated. Let $\mathbb{S} = \{(\mathsf{msg}, \mathsf{asig}) \mid \mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig}) = \perp\}$. Then our claim follows from the permissiveness of $\mathsf{pred}_{\mathsf{trivial}}$ because $\mathsf{L}^1_{\mathsf{aDec}} = \mathsf{L}^0_{\mathsf{aDec}} \setminus \mathbb{S}$, no elements of $\mathsf{L}_{\mathsf{aSign}}$ will be in $\mathbb{S}$, and the last element of $\mathsf{L}^1_{\mathsf{aDec}}$ will not be in $\mathbb{S}$. Hence,

$$\Pr[\mathcal{G}_0(\lambda) \Rightarrow 1] = \Pr[\mathcal{G}_1(\lambda) \Rightarrow 1].$$

Moving from game $\mathcal{G}_1$ to game $\mathcal{G}_2$, we switch from deciding the triviality of $\mathcal{A}$'s final output using the lists $\mathsf{L}_{\mathsf{aSign}}$ and $\mathsf{L}_{\mathsf{aDec}}$ which tracks the inputs and outputs of aS's algorithms to instead using lists $\mathsf{L}_{\mathsf{Enc}}$ and $\mathsf{L}_{\mathsf{Dec}}$ which track the inputs and outputs of prE. Define a function f by $f((\mathsf{msg}, \mathsf{asig})) = (\mathsf{msg}, \mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig}))$ when $\mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig}) \neq \perp$ and $f((\mathsf{msg}, \mathsf{asig})) = \perp$ otherwise. Because S is strongly randomness recovering, f is quasi-injective. By construction, $(\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}}) = (f(\mathsf{L}_{\mathsf{aSign}}), f(\mathsf{L}_{\mathsf{aDec}}))$ and $f(x) \neq \perp$ for all $x \in \mathsf{L}_{\mathsf{aSign}} \cup \mathsf{L}_{\mathsf{aDec}}$. The latter of these required our prior game transition. Thus, $\mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}}) = \mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}})$ because $\mathsf{pred}_{\mathsf{trivial}}$ is equality-pattern respecting. It follows that

$$\Pr[\mathcal{G}_1(\lambda) \Rightarrow 1] = \Pr[\mathcal{G}_2(\lambda) \Rightarrow 1].$$

The reduction adversary $\mathcal{B}$ that bounds the probability $\mathcal{A}$ wins game $\mathcal{G}_2$ is shown in Figure 22. It simulates the view of $\mathcal{A}$ by internally running the algorithms of S and querying its own oracles whenever algorithm of prE should be run. Suppose $\mathcal{A}$ halts and outputs $(\mathsf{msg}^*, \mathsf{asig}^*)$. The reduction will attempt to recover act^* from asig^*. If this fails, it aborts; otherwise, it returns $(\mathsf{msg}^*, \mathsf{act}^*)$ as its attempted forgery. Now $\mathcal{B}$ perfectly simulates the view of $\mathcal{A}$ in $\mathcal{G}_1$ and the lists $\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}}$ in $\mathcal{G}_2$ exactly match those in $\mathcal{B}$'s game, so $\mathcal{B}$ wins whenever $\mathcal{A}$ would, hence

$$\Pr[\mathcal{G}_1(\lambda) \Rightarrow 1] \leq \mathbf{Adv}^{\mathsf{INT\text{-}CT}}_{\mathsf{prE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{B}}(\lambda).$$

The full inequality in the theorem statement follows from the rearrangement and combination of the bounds on the transitions and the adversary's ability to win the final game. □

```
$\mathcal{G}_n(\lambda)$ for $n \in \{0,1,2\}$
  $\mathsf{rej} \leftarrow 0$
  $(\mathsf{L_{aSign}}, \mathsf{L_{aDec}}) \leftarrow ([\cdot],[\cdot])$   // $\mathcal{G}_{[0,2)}$
  $(\mathsf{L_{Enc}}, \mathsf{L_{Dec}}) \leftarrow ([\cdot],[\cdot])$   // $\mathcal{G}_{[2,\infty)}$
  $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
  $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
  $(\mathsf{k}, \mathsf{st_{Enc}}, \mathsf{st_{Dec}}) \leftarrow \mathsf{prE.KeyGen}(\mathsf{pp})$
  $(\mathsf{msg}^*, \mathsf{asig}^*) \leftarrow \mathcal{A}^{O_{\mathsf{aSign}}, O_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
  $\mathsf{act}^* \leftarrow \mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}^*, \mathsf{asig}^*)$
  if $\mathsf{act}^* = \bot \lor \mathsf{rej} = 1$ then return $0$
  $\mathsf{amsg}^* \leftarrow \mathsf{prE.Dec}(\mathsf{k}, \mathsf{msg}^*, \mathsf{act}^* : \mathsf{st_{Dec}})$
  $\mathsf{L_{aDec}}.\mathsf{add}((\mathsf{msg}^*, \mathsf{asig}^*))$   // $\mathcal{G}_{[0,2)}$
  $\mathsf{L_{Dec}}.\mathsf{add}((\mathsf{msg}^*, \mathsf{act}^*))$   // $\mathcal{G}_{[2,\infty)}$
  $b_{\mathsf{valid}} \leftarrow [\![\mathsf{amsg}^* \neq \bot]\!]$
  $b_{\mathsf{trivial}} \leftarrow \mathsf{pred}_{\mathsf{trivial}}(\mathsf{L_{aSign}}, \mathsf{L_{aDec}})$   // $\mathcal{G}_{[0,2)}$
  $b_{\mathsf{trivial}} \leftarrow \mathsf{pred}_{\mathsf{trivial}}(\mathsf{L_{Enc}}, \mathsf{L_{Dec}})$   // $\mathcal{G}_{[2,\infty)}$
  return $b_{\mathsf{valid}} \land \neg b_{\mathsf{trivial}}$
```

```
$O_{\mathsf{aSign}}(\mathsf{msg}, \mathsf{amsg})$
  $\mathsf{act} \leftarrow \mathsf{prE.Enc}(\mathsf{k}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st_{Enc}})$
  $\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; \mathsf{act})$
  $\mathsf{L_{aSign}}.\mathsf{add}((\mathsf{msg}, \mathsf{asig}))$   // $\mathcal{G}_{[0,2)}$
  $\mathsf{L_{Enc}}.\mathsf{add}((\mathsf{msg}, \mathsf{act}))$   // $\mathcal{G}_{[2,\infty)}$
  return $\mathsf{asig}$

$O_{\mathsf{aDec}}(\mathsf{msg}, \mathsf{asig})$
  $\mathsf{act} \leftarrow \mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig})$
  if $\mathsf{act} = \bot \lor \mathsf{rej} = 1$ then
    $\mathsf{rej} \leftarrow 1$   // strict
    $\mathsf{L_{aDec}}.\mathsf{add}((\mathsf{msg}, \mathsf{asig}))$   // $\mathcal{G}_{[0,1)}$
    return $\bot$
  $\mathsf{amsg} \leftarrow \mathsf{prE.Dec}(\mathsf{k}, \mathsf{msg}, \mathsf{act} : \mathsf{st_{Dec}})$
  $\mathsf{L_{aDec}}.\mathsf{add}((\mathsf{msg}, \mathsf{asig}))$   // $\mathcal{G}_{[0,2)}$
  $\mathsf{L_{Dec}}.\mathsf{add}((\mathsf{msg}, \mathsf{act}))$   // $\mathcal{G}_{[2,\infty)}$
  return $\mathsf{amsg}$
```

```
$\mathcal{B}^{O^{\mathcal{B}}_{\mathsf{Enc}}, O^{\mathcal{B}}_{\mathsf{Dec}}}(\mathsf{pp})$
  $\mathsf{rej} \leftarrow 0$
  $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
  $(\mathsf{msg}^*, \mathsf{asig}^*) \leftarrow \mathcal{A}^{O^{\mathcal{A}}_{\mathsf{aSign}}, O^{\mathcal{A}}_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
  $\mathsf{act}^* \leftarrow \mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}^*, \mathsf{asig}^*)$
  if $\mathsf{act}^* = \bot$ then abort
  return $(\mathsf{msg}^*, \mathsf{act}^*)$
```

```
$O^{\mathcal{A}}_{\mathsf{aSign}}(\mathsf{msg}, \mathsf{amsg})$
  $\mathsf{act} \leftarrow O^{\mathcal{B}}_{\mathsf{Enc}}(\mathsf{msg}, \mathsf{amsg})$
  $\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; \mathsf{act})$
  return $\mathsf{asig}$

$O^{\mathcal{A}}_{\mathsf{aDec}}(\mathsf{msg}, \mathsf{asig})$
  $\mathsf{act} \leftarrow \mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig})$
  if $\mathsf{act} = \bot \lor \mathsf{rej} = 1$ then
    $\mathsf{rej} \leftarrow 1$   // strict
    return $\bot$
  $\mathsf{amsg} \leftarrow O^{\mathcal{B}}_{\mathsf{Dec}}(\mathsf{msg}, \mathsf{act})$
  return $\mathsf{amsg}$
```

Fig. 22. Games (top) and reduction (bottom) used in the proof of Theorem 3

C.3 Full Proof of Theorem 4 (RIdP* is DUF-CASA)

In Section 4, we argued that the improved randomness identification with PRF transform RIdP* is dictator unforgeable. We provide a full proof here.

Proving DUF-CASA security is no easier than proving mc-DUF-CASA security (Appendix C.1), because we have to rule out forgeries being queried to the decryption oracle in both cases. So we prove the following result and Theorem 4 follows as a simple corollary from mc-DUF-CASA security implying DUF-CASA security as shown in Theorem 22.

Theorem 23 (RIdP* is mc-DUF-CASA). *Let* S *be a strongly randomness-identifying signature scheme and* prF *be pseudorandom function. Then* $\mathsf{aS} = \mathsf{RIdP}^\star[\mathsf{S}, \mathsf{prF}]$ *(Construction 2) is dictator unforgeable for* $\mathsf{pred}^{\mathsf{psync}}_{\mathsf{trivial}}$. *In particular, for all PPT adversaries $\mathcal{A}$ that make at most q_D anamorphic decryption queries* *and at most* $|\mathsf{aPKE}.\mathbb{ST}^{\mathsf{aEnc}}_{\mathsf{pp}}|$ *queries to either oracle, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}^{\mathsf{psync}}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{B}}(\lambda) + (q_D + 1)\frac{|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|}{|\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}|}.$$

The ideas underlying this proof are quite natural. Because S is strongly randomness-identifying, each $(\mathsf{msg}, \mathsf{asig})$ queried to anamorphic decryption corresponds to, at most, one randomness value, $r \leftarrow \mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{sig})$. To win at the game, the adversary will need one of its q_D queries to be a forgery corresponding to an r which is an output of the PRF not trivially forwarded from the signing oracle. There are at most $|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|$ PRF outputs that $r \in \mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}$ will be checked against in a single decryption query. Formalizing this takes a little care.

Proof. Let $\mathcal{A}$ be a PPT adversary against the $\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CASA}}$ game. We start with the PRF reduction $\mathcal{B}$ as defined in Figure 23. It simulates the $\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CASA}}$ game for $\mathcal{A}$ except replacing all uses of prF with queries to its oracle and returns 1 whenever $\mathcal{A}$ wins. Here **break** is an operation to leave the for loop early. Naturally, this reduction will transform $\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CASA}}$ to a version of the game where prF is replaced with a truly random function.

Game $\mathcal{G}_0$. The game $\mathcal{G}_0$ in Figure 23 is such a game, with some additional rewriting to prepare for future analysis. It was obtained by plugging the code of $\mathsf{RIdP}^{\star}[\mathsf{S}, \mathsf{prF}]$ and $\mathsf{pred}^{\mathsf{psync}}_{\mathsf{trivial}}$ into $\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CASA}}$ (with some minor editing for convenience). Then we replace prF with a random function $\mathcal{F}$ (where we let Funcs denote the appropriate set of functions for it to be sampled from). In the decryption oracle we rewrote it to compare $\mathcal{F}$ outputs with the possibly inefficient algorithm $\mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{sig})$ (from the strongly randomness-identifying property) and we wrote a separate case to set bad (but do nothing) when $(\mathsf{msg}, \mathsf{asig})$ was trivially forwarded from the signing oracle. We try to use $\mathsf{ctr}_{\mathsf{aDec}} - 1$ as the final counter that would be computed by $\mathsf{pred}^{\mathsf{psync}}_{\mathsf{trivial}}$, but if this is incorrect we set bad' and replace it with the correctly computed on from the algorithm CTR. We remove b_{valid} and simply return $\bot$ early if amsg is $\bot$. These modifications do not change the behavior of the game from $\mathcal{A}$'s perspective. So, we claim this is equivalent to the game that $\mathcal{B}$ transitions us to, meaning

$$\Pr[\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CASA}}_{\mathsf{aS},\mathsf{pred}^{\mathsf{psync}}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[\mathcal{G}_0(\lambda) \Rightarrow 1] = \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{A}}(\lambda).$$

Transition from game $\mathcal{G}_0$ to $\mathcal{G}_1$. In $\mathcal{G}_1$, when bad occurs we break out of the for loop early without incrementing $\mathsf{ctr}_{\mathsf{aDec}}$ or setting amsg to a non-$\bot$ value. Additionally, when bad' occurs we leave ctr equal to $\mathsf{ctr}_{\mathsf{aDec}} - 1$. Note that $\mathcal{G}_0$ and $\mathcal{G}_1$ are identical until one of the bad flags is set so

$$\Pr[\mathcal{G}_0(\lambda) \Rightarrow 1] - \Pr[\mathcal{G}_1(\lambda) \Rightarrow 1] \leq \Pr[\mathsf{bad} \vee \mathsf{bad}'].$$

We analyze these bad events in $\mathcal{G}_1$. To analyze this, we will first argue that bad' can only be set if bad was already set (which means $\Pr[\mathsf{bad} \vee \mathsf{bad}'] = \Pr[\mathsf{bad}]$). Then we bound the probability of bad.

Invariant analysis. We want as an invariant of $\mathcal{G}_1$ that if bad is false then bad' is false. To argue this it will be easier to argue the following invariant.

Invariant. *Before any oracle query, define* ctr *and* $i_1, \ldots, i_{\mathsf{ctr}-1}$ *as from running* $\mathsf{CTR}(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}})$*. If* bad *is false, then the* j*-th decryption query incremented* $\mathsf{ctr}_{\mathsf{aDec}}$ *if and only if* $j \in \{i_1, \ldots, i_{\mathsf{ctr}-1}\}$*.*

Note the invariant implies $\mathsf{ctr}_{\mathsf{aDec}} = \mathsf{ctr}$. In anamorphic decryption, if line 14 is reached then an additional entry was added to $\mathsf{L}_{\mathsf{aDec}}$ and $\mathsf{ctr}_{\mathsf{aDec}}$ was incremented. So at that time $\mathsf{ctr}_{\mathsf{aDec}} - 1$ is the value the counter had before the query and $\mathsf{L}_{\mathsf{aDec}}[1, \ldots, -2]$ is the value the list had before the query. Thus the invariant implies that if bad is false, $\mathsf{ctr} = \mathsf{ctr}'$ must hold and so bad' will not be set.

As a base case, suppose $|\mathsf{L}_{\mathsf{aSign}}| = 0$ or $|\mathsf{L}_{\mathsf{aDec}}| = 0$. In either case, $\mathsf{CTR}(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}})$ will never increment ctr, so $\{i_1, \ldots, i_{\mathsf{ctr}-1}\}$ is the trivially empty set. If $|\mathsf{L}_{\mathsf{aDec}}| = 0$, then no decryption queries have been made

```
$\mathcal{B}^{O^{\mathcal{B}}_{\mathsf{prF}}}(\mathsf{pp})$
$\mathsf{win} \leftarrow 0$
$(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}}) \leftarrow ([\cdot], [\cdot])$
$(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
$(\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{ctr}_{\mathsf{aDec}}) \leftarrow (1, 1)$
$\textbf{run } \mathcal{A}^{O_{\mathsf{aSign}}, O_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
$\textbf{return } [\![\mathsf{win}]\!]$
```

```
$O_{\mathsf{aSign}}(\mathsf{msg}, \mathsf{amsg})$
$r \leftarrow O^{\mathcal{B}}_{\mathsf{prF}}((\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{msg}, \mathsf{amsg}))$
$\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$
$\mathsf{ctr}_{\mathsf{aSign}} \leftarrow \mathsf{ctr}_{\mathsf{aSign}} + 1$
$\mathsf{L}_{\mathsf{aSign}}.\mathsf{add}((\mathsf{msg}, \mathsf{asig}))$
$\textbf{return } \mathsf{asig}$
```

```
$O_{\mathsf{aDec}}(\mathsf{msg}, \mathsf{asig})$
$\mathsf{amsg} \leftarrow \bot$
$\textbf{for } \mathsf{amsg}' \in \mathsf{aS}.\mathbb{AM}_{\mathsf{pp}} \textbf{ do}$
    $r' \leftarrow O^{\mathcal{B}}_{\mathsf{prF}}((\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{msg}, \mathsf{amsg}'))$
    $\textbf{if } \mathsf{S.RIdtfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{asig}, r') = 1 \textbf{ then}$
        $\mathsf{ctr}_{\mathsf{aDec}} \leftarrow \mathsf{ctr}_{\mathsf{aDec}} + 1$
        $\mathsf{amsg} \leftarrow \mathsf{amsg}'$
        $\textbf{break}$
$\mathsf{L}_{\mathsf{aDec}}.\mathsf{add}((\mathsf{msg}, \mathsf{asig}))$
$b_{\mathsf{valid}} \leftarrow [\![\mathsf{amsg} \neq \bot]\!]$
$b_{\mathsf{trivial}} \leftarrow \mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}})$
$\textbf{if } b_{\mathsf{valid}} \wedge \neg b_{\mathsf{trivial}} \textbf{ then } \mathsf{win} \leftarrow 1$
$\textbf{return } \mathsf{amsg}$
```

```
$\mathcal{G}_n(\lambda) \text{ for } n \in \{0, 1\}$
$\mathsf{win} \leftarrow 0$
$\mathcal{F} \leftarrow\!\!\$\ \mathsf{Funcs}$
$(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}}) \leftarrow ([\cdot], [\cdot])$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^{\lambda})$
$(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
$\mathsf{dk} \leftarrow \mathsf{prF.KeyGen}(\mathsf{pp})$
$(\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{ctr}_{\mathsf{aDec}}) \leftarrow (1, 1)$
$\textbf{run } \mathcal{A}^{O_{\mathsf{aSign}}, O_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
$\textbf{return } \mathsf{win}$
```

```
$O_{\mathsf{aSign}}(\mathsf{msg}, \mathsf{amsg})$
$r \leftarrow \mathcal{F}((\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{msg}, \mathsf{amsg}))$
$\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$
$\mathsf{ctr}_{\mathsf{aSign}} \leftarrow \mathsf{ctr}_{\mathsf{aSign}} + 1$
$\mathsf{L}_{\mathsf{aSign}}.\mathsf{add}((\mathsf{msg}, \mathsf{asig}))$
$\textbf{return } \mathsf{asig}$
```

```
$\mathsf{CTR}(\mathsf{L}_1, \mathsf{L}_2)$
$\mathsf{ctr} \leftarrow 1$
$\textbf{for } 1 \leq i \leq |\mathsf{L}_2| \textbf{ do}$
    $\textbf{if } \mathsf{L}_1[\mathsf{ctr}] = \mathsf{L}_2[i] \textbf{ then}$
        $i_{\mathsf{ctr}} \leftarrow i$
        $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
$\textbf{return } \mathsf{ctr}$
```

```
$O_{\mathsf{aDec}}(\mathsf{msg}, \mathsf{asig})$
$\mathsf{amsg} \leftarrow \bot$
$r \leftarrow \mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig})$
$\textbf{for } \mathsf{amsg}' \in \mathsf{aS}.\mathbb{AM}_{\mathsf{pp}} \textbf{ do}$
    $\textbf{if } r = \mathcal{F}((\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{msg}, \mathsf{amsg}')) \textbf{ then}$
        $\textbf{if } (\mathsf{msg}, \mathsf{asig}) \neq \mathsf{L}_{\mathsf{aSign}}[\mathsf{ctr}_{\mathsf{aDec}}] \textbf{ then}$
            $\mathsf{bad} \leftarrow \mathsf{true}$
            $\textbf{break} \quad /\!\!/ \ \mathcal{G}_{[1,\infty)}$
        $\mathsf{ctr}_{\mathsf{aDec}} \leftarrow \mathsf{ctr}_{\mathsf{aDec}} + 1$
        $\mathsf{amsg} \leftarrow \mathsf{amsg}'$
        $\textbf{break}$
$\mathsf{L}_{\mathsf{aDec}}.\mathsf{add}((\mathsf{msg}, \mathsf{asig}))$
$\textbf{if } \mathsf{amsg} = \bot \textbf{ then return } \bot$
$\mathsf{ctr} \leftarrow \mathsf{ctr}_{\mathsf{aDec}} - 1$
$\mathsf{ctr}' \leftarrow \mathsf{CTR}(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_{\mathsf{aDec}}[1, \ldots, -2])$
$\textbf{if } \mathsf{ctr} \neq \mathsf{ctr}' \textbf{ then}$
    $\mathsf{bad}' \leftarrow \mathsf{true}$
    $\mathsf{ctr} \leftarrow \mathsf{ctr}' \quad /\!\!/ \ \mathcal{G}_{[0,1)}$
$b_{\mathsf{trivial}} \leftarrow [\![\mathsf{L}_{\mathsf{aSign}}[\mathsf{ctr}] = (\mathsf{msg}, \mathsf{asig})]\!]$
$\textbf{if } \neg b_{\mathsf{trivial}} \textbf{ then } \mathsf{win} \leftarrow 1$
$\textbf{return } \mathsf{amsg}$
```

Fig. 23. Reduction (top) and games (bottom) used in the proof of Theorem 23

and clearly none of them incremented $\mathsf{ctr}_{\mathsf{aDec}}$. If $|\mathsf{L}_{\mathsf{aSign}}| = 0$, no decryption queries will have incremented $\mathsf{ctr}_{\mathsf{aDec}}$, because they can only do so when $(\mathsf{msg}, \mathsf{asig}) = \mathsf{L}_{\mathsf{aSign}}[\mathsf{ctr}_{\mathsf{aDec}}]$.

Now we inductively argue that if the invariant holds before an oracle query, then it still holds after the oracle query. We only need to consider the case when bad is false after the oracle query as the invariant clearly holds whenever bad is true.

Consider an anamorphic signing query. It cannot increment $\mathsf{ctr}_{\mathsf{aDec}}$, so it could only make the invariant false only if it changed the set $\{i_1, \ldots, i_{\mathsf{ctr}-1}\}$. Let L_1 and L_1' denote the values of $\mathsf{L}_{\mathsf{aSign}}$ before and after the query. The computations $\mathsf{CTR}(\mathsf{L}_1, \mathsf{L}_{\mathsf{aDec}})$ and $\mathsf{CTR}(\mathsf{L}_1', \mathsf{L}_{\mathsf{aDec}})$ can only differ once $\mathsf{ctr} = |\mathsf{L}_1'|$, as only then is the new entry of L_1' used. This is the same value $\mathsf{ctr}_{\mathsf{aSign}}$ held at the beginning of the oracle query. For the computation to differ then it requires that $\mathsf{L}_{\mathsf{aDec}}[i] = \mathsf{L}_1'[\mathsf{ctr}]$ for some $i \in \{i_{\mathsf{ctr}-1} + 1, \ldots, |\mathsf{L}_{\mathsf{aDec}}|\}$. By the invariant holding before the query, it must be that $\mathsf{ctr}_{\mathsf{aDec}} = \mathsf{ctr}$ held at beginning of the i-th decryption query. But then the fact that $\mathsf{L}_1'[\mathsf{ctr}] = (\mathsf{msg}, \mathsf{asig})$ implies that asig is a signature of msg using randomness $\mathcal{F}((\mathsf{ctr}, \mathsf{msg}, \mathsf{amsg}))$ for the amsg queried the anamorphic signing oracle. Consequently line 4 would have evaluated true at some point in the i-th decryption query (not necessarily with amsg') which would have set bad (as $\mathsf{L}_{\mathsf{aSign}}[\mathsf{ctr}] = \perp$ at that time). This contradicts our assumption that bad is false.

Consider the j-th anamorphic decryption query. Let L_2 and L_2' denote the values of $\mathsf{L}_{\mathsf{aDec}}$ before and after the query. Note $j = |\mathsf{L}_2'|$. Let l denote the value of $\mathsf{ctr}_{\mathsf{aDec}}$ before the query. By the invariant, $l = \mathsf{CTR}(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_2)$. The computations $\mathsf{CTR}(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_2)$ and $\mathsf{CTR}(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_2')$ differ only in the latter having one more iteration of the for loop, which will change the outcome of the computation if and only if $\mathsf{L}_{\mathsf{aSign}}[l] = \mathsf{L}_2'[|\mathsf{L}_2'|]$, which would result in $i_l \leftarrow |\mathsf{L}_2'|$ and ctr being incremented one more time.

First suppose that this query increments $\mathsf{ctr}_{\mathsf{aDec}}$ from l to $l+1$. This requires $\mathsf{L}_2'[-1] = \mathsf{L}_{\mathsf{aSign}}[l]$, otherwise it would have broken out of the for loop on line 7 before incrementing $\mathsf{ctr}_{\mathsf{aDec}}$. But then as discussed above, $i_l \leftarrow |\mathsf{L}_2'|$ would occur in $\mathsf{CTR}(\mathsf{L}_{\mathsf{aSign}}, \mathsf{L}_2')$. Hence the invariant would still hold.

In the other direction, suppose this query changes the set $\{i_1, \ldots, i_{\mathsf{ctr}-1}\}$. As discussed above, this is only possible if $\mathsf{L}_{\mathsf{aSign}}[l] = \mathsf{L}_2'[|\mathsf{L}_2'|]$ and results in $i_l \leftarrow |\mathsf{L}_2'|$ being added to the set. The entry $\mathsf{L}_{\mathsf{aSign}}[l]$ was added using $\mathsf{ctr}_{\mathsf{aSign}} = l$, then because the randomness it used must be the randomness recovered in the current anamorphic decryption query, at some point line 4 and not line 5 must have evaluated to true. This results in $\mathsf{ctr}_{\mathsf{aDec}}$ being incremented. Hence the invariant would still hold.

Bounding the bad event. Now we bound $\Pr[\mathsf{bad}]$ in $\mathcal{G}_1$. First note that when bad is set, the tuple $(\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{msg}, \mathsf{amsg}')$ cannot previously have been used as input to $\mathcal{F}$ while anamorphically signing. If it was, then $\mathsf{L}_{\mathsf{aSign}}[\mathsf{ctr}_{\mathsf{aDec}}]$ would equal $(\mathsf{msg}, \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r))$. But the correctness of $\mathsf{S.RRecov}$ implies that $\mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)) = r$ and because $\mathsf{S.RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \cdot)$ is a quasi-injection, r can only have one pre-image. Thus, $\mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r) = \mathsf{asig}$. Then line 5 would have been false and bad would not been set.

So when bad is set, $\mathcal{F}((\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{msg}, \mathsf{amsg}'))$ could only previously have been used in line 4. Outputs of $\mathcal{F}$ are random from a set of size at least $|\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}|$. For each of the q_D decryption queries, there are at most $|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|$ outputs of $\mathcal{F}$ that r could equal to set bad. (There's at most $|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}| - 1$ if a $\mathcal{F}((\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{msg}, \cdot))$ computation has been used while anamorphically signing. If there are collisions in $\mathcal{F}((\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{msg}, \cdot))$ it can be fewer than $|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}| - 1$.) Thus standard analysis gives

$$\Pr[\mathsf{bad}] \leq q_D \frac{|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|}{|\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}|}.$$

Game $\mathcal{G}_1$. Finally, we conclude the proof by noting that

$$\Pr[\mathcal{G}_1(\lambda) \Rightarrow 1] = 0.$$

The flag win can only be set on line 19. This require $\mathsf{L}_{\mathsf{aSign}}[\mathsf{ctr}] \neq (\mathsf{msg}, \mathsf{asig})$ (from $\neg b_{\mathsf{trivial}}$) and $\mathsf{amsg} \neq \perp$ (from line 12). The latter could only occur on line 9, but then the former would have resulted in breaking out of the for loop already on line 7. The bound follows by combining our claims. □

D Porting Results (Back) to Anamorphic Encryption

Prior to this point we considered dictators who wish to forge anamorphic *signatures* that contain valid anamorphic messages within them and addressed gaps in a robustness notion of BGHMR [BGH+24]. To present our dictator unforgeability definition, we first adapted BGHMR's results to the signature scheme setting because they presented their results in the context of anamorphic *encryption*. However, our definitions, proofs, and attacks did not leverage any particular property of signature schemes and hence also apply to anamorphic encryption with little modification. This is unsurprising as the dictator in the anamorphic threat model receives both public encryption/verification key and secret decryption/signing key, so the two primitives are effectively the same from the dictator's perspective. In this section, we give relevant background, formalize dictator unforgeability for anamorphic encryption, and argue that our results for anamorphic signature schemes apply directly to the encryption setting as well.

D.1 Syntax and Security of Anamorphic Encryption

Anamorphic encryption [PPY22] (PPY) operates similarly to anamorphic signature schemes in that the objective of anamorphic encryption is to appear like regular public-key encryption except that a user knowing a double key dk can encrypt a covert message into a ciphertext act. This covert message, called the anamorphic message amsg, can be recovered from act using the secret key sk and dk.[5] Anamorphic encryption is meant to be stealthy under dictators who may coerce parties into revealing their public encryption and secret decryption keys.

Definition 30 (Anamorphic Encryption Scheme). *An* anamorphic encryption scheme aPKE *is a public-key encryption scheme (specifying* $\mathsf{aPKE.KeyGen}$*,* $\mathsf{aPKE.Enc}$*, and* $\mathsf{aPKE.Dec}$*) with three additional PPT algorithms.*

- $\mathsf{aPKE.aKeyGen(pp)}$ *takes public parameters* pp *and generates the secret key* $\mathsf{sk} \in \mathsf{aPKE}.\mathbb{SK}_{\mathsf{pp}}$*, public key* $\mathsf{pk} \in \mathsf{aPKE}.\mathbb{PK}_{\mathsf{pp}}$*, double key* $\mathsf{dk} \in \mathsf{aPKE}.\mathbb{DK}_{\mathsf{pp}}$*, initial anamorphic encryption state* $\mathsf{st}_{\mathsf{aEnc}} \in \mathsf{aPKE}.\mathbb{ST}^{\mathsf{aEnc}}_{\mathsf{pp}}$*, and initial anamorphic decryption state* $\mathsf{st}_{\mathsf{aDec}} \in \mathsf{aPKE}.\mathbb{ST}^{\mathsf{aDec}}_{\mathsf{pp}}$*. It outputs* $(\mathsf{pk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aEnc}}, \mathsf{st}_{\mathsf{aDec}})$*.*
- $\mathsf{aPKE.aEnc}(\mathsf{pk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aEnc}})$ *takes the public key* $\mathsf{pk} \in \mathsf{aPKE}.\mathbb{PK}_{\mathsf{pp}}$*, double key* $\mathsf{dk} \in \mathsf{aPKE}.\mathbb{DK}_{\mathsf{pp}}$*, message* $\mathsf{msg} \in \mathsf{aPKE}.\mathbb{M}_{\mathsf{pp}}$*, anamorphic message* $\mathsf{amsg} \in \mathsf{aPKE}.\mathbb{AM}_{\mathsf{pp}}$*, and anamorphic encryption state* $\mathsf{st}_{\mathsf{aEnc}} \in \mathsf{aPKE}.\mathbb{ST}^{\mathsf{aEnc}}_{\mathsf{pp}}$ *and outputs an anamorphic ciphertext* $\mathsf{act} \in \mathsf{aPKE}.\mathbb{AC}_{\mathsf{pp}}$*. It also updates the state* $\mathsf{st}_{\mathsf{aEnc}}$*.*
- $\mathsf{aPKE.aDec}(\mathsf{sk}, \mathsf{dk}, \mathsf{act} : \mathsf{st}_{\mathsf{aDec}})$ *takes the secret key* $\mathsf{sk} \in \mathsf{aPKE}.\mathbb{SK}_{\mathsf{pp}}$*, double key* $\mathsf{dk} \in \mathsf{aPKE}.\mathbb{DK}_{\mathsf{pp}}$*, anamorphic ciphertext* $\mathsf{act} \in \mathsf{aPKE}.\mathbb{AC}_{\mathsf{pp}}$*, and anamorphic decryption state* $\mathsf{st}_{\mathsf{aDec}} \in \mathsf{aPKE}.\mathbb{ST}^{\mathsf{aDec}}_{\mathsf{pp}}$ *and outputs an anamorphic message* $\mathsf{amsg} \in \mathsf{aPKE}.\mathbb{AM}_{\mathsf{pp}}$*. It also updates the state* $\mathsf{st}_{\mathsf{aDec}}$*.*

Anamorphic encryption schemes aim to be correct, stealthy, and confidential with respect to the anamorphic messages. This is captured by the $\mathcal{G}^{\mathsf{CORR}}$, $\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}$, $\mathcal{G}^{\mathsf{IND\text{-}CAMA}}$, and $\mathcal{G}^{\mathsf{IND\text{-}CACA}}$ games in Figure 24 and the following four definitions. The first three were introduced in PPY and the last is again a natural extension we provide.

Definition 31 (Correctness). *An anamorphic encryption scheme* aPKE *is* correct *if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{aPKE},\mathcal{A}}(\lambda)$ *is negligible where*

$$\mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) \Rightarrow 1].$$

Definition 32 (Stealthiness). *An anamorphic encryption scheme* aPKE *is stealthy (in the* $\mathsf{RoA\text{-}CAMA}$ *sense) if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda)$ *is negligible where*

$$\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) = 2\Pr[\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) \Rightarrow 1] - 1.$$

[5] Some anamorphic encryption schemes does not require sk when anamorphically decrypting [BGH+24], i.e., anyone who knows dk can extract amsg from act. We do not syntactically differentiate this type of scheme.

```
$\mathcal{G}^{\mathsf{CORR}}_{\mathsf{aPKE},\mathcal{A}}(\lambda)$
$\mathsf{win} \leftarrow 0$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aEnc}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow \mathsf{aPKE.KeyGen}(\mathsf{pp})$
$\textbf{run } \mathcal{A}^{O_{\mathsf{aSign},\mathsf{aDec}}}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk})$
$\textbf{return } \mathsf{win}$
```

```
$O_{\mathsf{aEnc},\mathsf{aDec}}(\mathsf{msg}, \mathsf{amsg})$
$\mathsf{act} \leftarrow \mathsf{aPKE.aEnc}(\mathsf{pk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aEnc}})$
$\mathsf{amsg}^* \leftarrow \mathsf{aPKE.aDec}(\mathsf{sk}, \mathsf{dk}, \mathsf{act} : \mathsf{st}_{\mathsf{aDec}})$
$\mathsf{win} \leftarrow \mathsf{win} \vee [\![\mathsf{amsg} \neq \mathsf{amsg}^*]\!]$
$\textbf{return } \mathsf{act}$
```

```
$\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda)$
$b \leftarrow\!\!\$\ \{0, 1\}$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
$\textbf{if } b = 0 \textbf{ then}$
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{aPKE.KeyGen}(\mathsf{pp})$
$\textbf{else}$
    $(\mathsf{pk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aEnc}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow \mathsf{aPKE.aKeyGen}(\mathsf{pp})$
$b^* \leftarrow \mathcal{A}^{O_{\mathsf{Enc}/\mathsf{aEnc}}}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk})$
$\textbf{return } [\![b = b^*]\!]$
```

```
$O_{\mathsf{Enc}/\mathsf{aEnc}}(\mathsf{msg}, \mathsf{amsg})$
$\textbf{if } b = 0 \textbf{ then}$
    $\mathsf{ct} \leftarrow \mathsf{aPKE.Enc}(\mathsf{pk}, \mathsf{msg})$
    $\textbf{return } \mathsf{ct}$
$\textbf{else}$
    $\mathsf{act} \leftarrow \mathsf{aPKE.aEnc}(\mathsf{pk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aEnc}})$
    $\textbf{return } \mathsf{act}$
```

```
$\mathcal{G}^{\mathsf{IND\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda)$ $\mathcal{G}^{\mathsf{IND\text{-}CACA}}_{\mathsf{aPKE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda)$
$b \leftarrow\!\!\$\ \{0, 1\}$
$(\mathsf{L}_{\mathsf{aEnc}}, \mathsf{L}_{\mathsf{aDec}}) \leftarrow ([\cdot], [\cdot])$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aEnc}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow \mathsf{aPKE.aKeyGen}(\mathsf{pp})$
$(\mathsf{msg}^*, \mathsf{amsg}^*_0, \mathsf{amsg}^*_1, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{O_{\mathsf{aEnc}}, O_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk})$
$\mathsf{act}^* \leftarrow \mathsf{aPKE.aEnc}(\mathsf{pk}, \mathsf{dk}, \mathsf{msg}^*, \mathsf{amsg}^*_b : \mathsf{st}_{\mathsf{aEnc}})$
$\mathsf{L}_{\mathsf{aEnc}}.\mathsf{add}((\mathsf{msg}^*, \mathsf{act}^*))$
$b^* \leftarrow \mathcal{A}^{O_{\mathsf{aEnc}}, O_{\mathsf{aDec}}}(\mathsf{st}_{\mathcal{A}}, \mathsf{act}^*)$
$\textbf{return } [\![b = b^*]\!]$
```

```
$O_{\mathsf{aEnc}}(\mathsf{msg}, \mathsf{amsg})$
$\mathsf{act} \leftarrow \mathsf{aPKE.aEnc}(\mathsf{pk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aEnc}})$
$\mathsf{L}_{\mathsf{aEnc}}.\mathsf{add}((\mathsf{msg}, \mathsf{act}))$
$\textbf{return } \mathsf{act}$
```

```
$O_{\mathsf{aDec}}(\mathsf{act})$
$\mathsf{msg} \leftarrow \mathsf{aPKE.Dec}(\mathsf{sk}, \mathsf{act})$
$\mathsf{amsg} \leftarrow \mathsf{aPKE.aDec}(\mathsf{sk}, \mathsf{dk}, \mathsf{act} : \mathsf{st}_{\mathsf{aDec}})$
$\mathsf{L}_{\mathsf{aDec}}.\mathsf{add}((\mathsf{msg}, \mathsf{act}))$
$\textbf{if } \mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{aEnc}}, \mathsf{L}_{\mathsf{aDec}}) = 1 \textbf{ then return } \bot$
$\textbf{return } \mathsf{amsg}$
```

Fig. 24. Anamorphic encryption correctness game $\mathcal{G}^{\mathsf{CORR}}$ (top), stealthiness game $\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}$ (middle), and confidentiality games $\mathcal{G}^{\mathsf{IND\text{-}CAMA}}$ and $\mathcal{G}^{\mathsf{IND\text{-}CACA}}$ (bottom)

Definition 33 (IND-CAMA Confidentiality). *An anamorphic encryption scheme* aPKE *is indistinguishable (in the* IND-CAMA *sense) if, for all PPT adversaries $\mathcal{A}$, the function $\mathbf{Adv}^{\mathsf{IND\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda)$ is negligible where*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) = 2 \Pr[\mathcal{G}^{\mathsf{IND\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) \Rightarrow 1] - 1.$$

Definition 34 (IND-CACA Confidentiality). *An anamorphic signature scheme* aPKE *is indistinguishable (in the* IND-CACA *sense) for a triviality predicate* $\mathsf{pred}_{\mathsf{trivial}}$ *if, for all PPT adversaries $\mathcal{A}$, the function $\mathbf{Adv}^{\mathsf{IND\text{-}CACA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda)$ is negligible where*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CACA}}_{\mathsf{aPKE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) = 2 \Pr[\mathcal{G}^{\mathsf{IND\text{-}CACA}}_{\mathsf{aPKE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \Rightarrow 1] - 1.$$

Akin to the result for anamorphic signatures (Theorem 1), PPY show that (RoA-CAMA) stealthiness implies (IND-CAMA) anamorphic message confidentiality for anamorphic encryption.

```
$\mathsf{aS}.\mathbb{DK}_{\mathsf{pp}} = \mathsf{prE}.\mathbb{K}_{\mathsf{pp}}$
$\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}} = \mathsf{prE}.\mathbb{M}_{\mathsf{pp}}$
$\mathsf{aS}.\mathbb{ST}^{\mathsf{aEnc}}_{\mathsf{pp}} = \mathsf{prE}.\mathbb{ST}^{\mathsf{Enc}}_{\mathsf{pp}}$
$\mathsf{aS}.\mathbb{ST}^{\mathsf{aDec}}_{\mathsf{pp}} = \mathsf{prE}.\mathbb{ST}^{\mathsf{Dec}}_{\mathsf{pp}}$

$\mathsf{aPKE.KeyGen} = \mathsf{PKE.KeyGen}$
$\mathsf{aPKE.Enc} = \mathsf{PKE.Enc}$
$\mathsf{aPKE.Dec} = \mathsf{PKE.Dec}$

$\mathsf{aPKE.aKeyGen}(\mathsf{pp})$
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp})$
  $(\mathsf{k}, \mathsf{st}_{\mathsf{Enc}}, \mathsf{st}_{\mathsf{Dec}}) \leftarrow \mathsf{prE.KeyGen}(\mathsf{pp})$
  $\mathsf{dk} \leftarrow (\mathsf{pk}, \mathsf{k})$
  $(\mathsf{st}_{\mathsf{aEnc}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow (\mathsf{st}_{\mathsf{Enc}}, \mathsf{st}_{\mathsf{Dec}})$
  return $(\mathsf{pk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aEnc}}, \mathsf{st}_{\mathsf{aDec}})$
```

```
$\mathsf{aPKE.aEnc}(\mathsf{sk}, \mathsf{dk} = (\mathsf{pk}, \mathsf{k}), \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aEnc}})$
  $r \leftarrow \mathsf{prE.Enc}(\mathsf{k}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aEnc}})$
  $\mathsf{act} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}; r)$
  return $\mathsf{act}$

$\mathsf{aPKE.aDec}(\mathsf{sk}, \mathsf{dk} = (\mathsf{pk}, \mathsf{k}), \mathsf{act} : \mathsf{st}_{\mathsf{aDec}})$
  $\mathsf{msg} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{act})$
  $r \leftarrow \mathsf{PKE.RRecov}(\mathsf{pk}, \mathsf{sk}, \mathsf{act})$
  if $\mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}; r) \neq \mathsf{act}$ then
    return $\perp$
  $\mathsf{amsg} \leftarrow \mathsf{prE.Dec}(\mathsf{k}, \mathsf{msg}, r : \mathsf{st}_{\mathsf{aDec}})$
  return $\mathsf{amsg}$
```

Fig. 25. Randomness replacement transforms RRep [KPP+23b] and RRep*

Theorem 24 (Theorem 1 of [PPY22]). *Let* aPKE *be a stealthy anamorphic encryption scheme. Then it is also* IND-CAMA *indistinguishable. In particular, for all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) \leq 2\,\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{B}}(\lambda).$$

D.2 Constructing Anamorphic Encryption

In this section, we outline the anamorphic encryption version of RRep proposed in [KPP+23b] and the original RIdP and RIdPX transforms proposed in BGHMR. For each, we discuss correctness and stealthiness. We also specify and discuss our modified transforms RRep* and RIdP*.

Anamorphism via Randomness Replacement. We begin by defining randomness recovery for public-key encryption, which is needed to construct anamorphic encryption schemes via RRep and RRep*.

Definition 35 (Randomness Recovery). *A public-key encryption scheme* PKE *is* randomness recovering *if it additionally specifies a PPT algorithm* PKE.RRecov *such that, for all* $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp})$, $\mathsf{msg} \in \mathsf{PKE}.\mathbb{M}_{\mathsf{pp}}$, $r \in \mathsf{PKE}.\mathbb{R}^{\mathsf{Enc}}_{\mathsf{pp}}$, *and* $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}; r)$, *$r$ can be recovered by computing* $r \leftarrow \mathsf{PKE.RRecov}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct})$.

We will not consider public randomness recovering schemes which do not require the use of sk to recover r, as a scheme allowing this would not be IND-CPA secure. Rather than define strong randomness recovery for public-key encryption schemes, we will instead define schemes to explicitly check that encrypting a message with the recovered randomness gives the specified ciphertext where appropriate.

Construction 3 (RRep and RRep*). *Consider the following primitives and requirements needed to construct an anamorphic encryption scheme via randomness replacement.*

- *Let* PKE *be a randomness-recovering public-key encryption scheme with recovery function* PKE.RRecov.
- *Let* prE *be a pseudorandom encryption scheme supporting associated data.*
- *Let* PublicParamGen *output parameters* pp *such that* $\mathsf{PKE}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}} = \mathsf{prE}.\mathbb{C}_{\mathsf{pp}}$ *and* $\mathsf{PKE}.\mathbb{M}_{\mathsf{pp}} = \mathsf{prE}.\mathbb{AD}_{\mathsf{pp}}$.

Then the anamorphic encryption schemes aPKE = RRep[PKE, prE] *(no highlighted parts) and* aPKE = RRep*[PKE, prE] *(with highlighted parts) are constructed as shown in Figure 25.*

The following theorems capture that RRep is correct and stealthy, as discussed in [KPP+23b]. We extend these results to $\mathsf{RRep}^\star$.

Theorem 25 (RRep is Correct). *Let* PKE *be a randomness-recovering public-key encryption scheme and* prE *be a pseudorandom encryption scheme. Then* $\mathsf{aPKE} = \mathsf{RRep}[\mathsf{PKE}, \mathsf{prE}]$ *(Construction 3) is correct. In particular, for all PPT adversaries* $\mathcal{A}$*, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{prE},\mathcal{B}}(\lambda).$$

Theorem 26 (Lemma 1 of [KPP+23b]). *Let* PKE *be a randomness-recovering public-key encryption scheme and* prE *be a pseudorandom encryption scheme. Then* $\mathsf{aPKE} = \mathsf{RRep}[\mathsf{PKE}, \mathsf{prE}]$ *(Construction 1) is stealthy. In particular, for all PPT adversaries* $\mathcal{A}$*, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{prE},\mathcal{B}}(\lambda).$$

Theorem 27 (RRep* is Correct). *Let* PKE *be a correct randomness-recovering public-key encryption scheme and* prE *be a pseudorandom encryption scheme. Then* $\mathsf{aPKE} = \mathsf{RRep}^\star[\mathsf{PKE}, \mathsf{prE}]$ *(Construction 3) is correct. In particular, for all PPT adversaries* $\mathcal{A}$*, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{prE},\mathcal{B}}(\lambda).$$

Proof. The proof follows similarly to the proof of Theorem 12. Because $\mathsf{RRep}^\star$ requires that the underlying PKE can recover randomness encryption perfectly and PKE is perfectly correct, any anamorphic encryption that breaks correctness must break the correctness of the underlying pseudorandom encryption scheme prE. Given a PPT adversary $\mathcal{A}$ against the correctness of aPKE, we can construct a PPT reduction $\mathcal{B}$ against the correctness of prE that internally generates the PKE keypair and performs encryption and decryption seeded with randomness from the prE encryption oracle $\mathcal{B}$ accesses. □

Theorem 28 (RRep* is RoA-CAMA). *Let* PKE *be a randomness-recovering public-key encryption scheme and* prE *be a pseudorandom encryption scheme. Then* $\mathsf{aPKE} = \mathsf{RRep}^\star[\mathsf{PKE}, \mathsf{prE}]$ *(Construction 1) is stealthy. In particular, for all PPT adversaries* $\mathcal{A}$*, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{prE},\mathcal{B}}(\lambda).$$

Proof. The proof follows identically to the proof of Theorem 13. Recall that $\mathsf{RRep}^\star$ replaces the encryption randomness with pseudorandom encryption scheme ciphertexts which appears uniformly random to an adversary that does not know the symmetric key, which is the $\mathsf{RRep}^\star$ double key. Given a PPT distinguishing adversary $\mathcal{A}$ against the $\mathsf{RoA\text{-}CAMA}$ security of aPKE, we can construct PPT reduction $\mathcal{B}$ against the $\mathsf{IND\$\text{-}CPA}$ security of prE that internally generates the PKE keypair and performs encryption and decryption seeded with randomness from the prE encryption oracle $\mathcal{B}$. □

Anamorphism via Randomness Identification with PRF. We now state the original randomness identification with PRF transform RIdP and randomness identification with PRF and XOR transform RIdPX from BGHMR. These are referred as Σ_1 and Σ_1' respectively in BGHMR. We also state the improved $\mathsf{RIdP}^\star$ transform. These transforms take in a public-key encryption scheme PKE and pseudorandom function prF and output an anamorphic encryption scheme aPKE. For correctness, we require that the underlying PKE has an injective randomness property.

Definition 36 (Injective Randomness). *A public-key encryption scheme* PKE *has* injective randomness *if* $\mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}; r) \neq \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}; r')$ *for all* $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$*,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp})$*,* $\mathsf{msg} \in \mathsf{PKE}.\mathbb{M}_{\mathsf{pp}}$*, and* $r \neq r' \in \mathsf{PKE}.\mathbb{R}^{\mathsf{Enc}}_{\mathsf{pp}}$.

Construction 4. *Consider the following requirements needed to construct an anamorphic encryption scheme via randomness identification with PRF (and XOR).*

- *Let* PKE *be a public-key scheme with injective randomness.*

```
$\mathsf{aPKE}.\mathbb{DK}_{\mathsf{pp}} = \mathsf{aPKE}.\mathbb{PK}_{\mathsf{pp}} \times \mathsf{prF}.\mathbb{K}_{\mathsf{pp}}$
$\boxed{\mathsf{aPKE}.\mathbb{AM}_{\mathsf{pp}} = \mathsf{prF}.\mathbb{R}_{\mathsf{pp}}}$
$\mathsf{aPKE}.\mathbb{ST}^{\mathsf{aEnc}}_{\mathsf{pp}} = \mathsf{aPKE}.\mathbb{ST}^{\mathsf{aDec}}_{\mathsf{pp}} = \mathbb{Z}_n$

$\mathsf{aPKE.KeyGen} = \mathsf{PKE.KeyGen}$
$\mathsf{aPKE.Enc} = \mathsf{PKE.Enc}$
$\mathsf{aPKE.Dec} = \mathsf{PKE.Dec}$

$\mathsf{aPKE.aKeyGen}(\mathsf{pp})$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp})$
$\mathsf{k} \leftarrow \mathsf{prF.KeyGen}(\mathsf{pp})$
$\mathsf{dk} \leftarrow (\mathsf{pk}, \mathsf{k})$
$\mathsf{ctr}_{\mathsf{aEnc}} \leftarrow 1$
$\mathsf{ctr}_{\mathsf{aDec}} \leftarrow 1$
return $(\mathsf{pk}, \mathsf{sk}, \mathsf{dk}, \mathsf{ctr}_{\mathsf{aEnc}}, \mathsf{ctr}_{\mathsf{aDec}})$

$\mathsf{aPKE.aEnc}(\mathsf{pk}, \mathsf{dk} = (\mathsf{pk}, \mathsf{k}), \mathsf{msg}, \mathsf{amsg} : \mathsf{ctr}_{\mathsf{aEnc}})$
$r \leftarrow \boxed{\mathsf{prF}(\mathsf{k}, (\mathsf{ctr}_{\mathsf{aEnc}}, \mathsf{msg}, \mathsf{amsg}))}$
      $\boxed{\mathsf{amsg} \oplus \mathsf{prF}(\mathsf{k}, \mathsf{ctr}_{\mathsf{aEnc}})}$
$\mathsf{act} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}; r)$
$\mathsf{ctr}_{\mathsf{aEnc}} \leftarrow \mathsf{ctr}_{\mathsf{aEnc}} + 1$
return $\mathsf{act}$

$\mathsf{aPKE.aDec}(\mathsf{sk}, \mathsf{dk} = (\mathsf{pk}, \mathsf{k}), \mathsf{act} : \mathsf{ctr}_{\mathsf{aDec}})$
$\mathsf{msg} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{act})$
for $\mathsf{amsg} \in \mathsf{aPKE}.\mathbb{AM}_{\mathsf{pp}}$ do
    $r' \leftarrow \boxed{\mathsf{prF}(\mathsf{k}, (\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{msg}, \mathsf{amsg}))}$
          $\boxed{\mathsf{amsg} \oplus \mathsf{prF}(\mathsf{k}, \mathsf{ctr}_{\mathsf{aDec}})}$
    if $\mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}; r') = \mathsf{act}$ then
        $\mathsf{ctr}_{\mathsf{aDec}} \leftarrow \mathsf{ctr}_{\mathsf{aDec}} + 1$
        return $\mathsf{amsg}$
return $\perp$
```

Fig. 26. Randomness identification transforms $\boxed{\mathsf{RIdP}}$, $\boxed{\mathsf{RIdP}^\star}$, [BGH+24] and $\boxed{\mathsf{RIdPX}}$

- *Let* prF *be a pseudorandom function that takes in* $\boxed{\textit{2-tuple}}$, $\boxed{\textit{3-tuple}}$, *or* $\boxed{\textit{1-tuple}}$ *messages where the first element of the tuple is an integer.*
- *Let* PublicParamGen *output parameters* pp *such that* $\mathsf{PKE}.\mathbb{R}^{\mathsf{Enc}}_{\mathsf{pp}} = \mathsf{prF}.\mathbb{R}_{\mathsf{pp}}$ *where these sets form a group over* $\oplus$ *and* $\mathsf{prF}.\mathbb{M}_{\mathsf{pp}} = \boxed{\boxed{\mathbb{Z}_n} \times \mathsf{PKE}.\mathbb{M}_{\mathsf{pp}} \times \mathsf{aPKE}.\mathbb{AM}_{\mathsf{pp}}}$ *where n is a positive integer defined in* pp.

Then the anamorphic encryption schemes $\boxed{\mathsf{aPKE} = \mathsf{RIdP}[\mathsf{PKE}, \mathsf{prF}]}$ *and* $\boxed{\mathsf{aPKE} = \mathsf{RIdPX}[\mathsf{PKE}, \mathsf{prF}]}$ *(no highlighted parts) and* $\boxed{\mathsf{aPKE} = \mathsf{RIdP}^\star[\mathsf{PKE}, \mathsf{prF}]}$ *(with highlighted parts) are constructed as shown in Figure 26.*

In all theorems, we will assume that $|\mathsf{aPKE}.\mathbb{AM}_{\mathsf{pp}}|$ is polynomially upper bounded and that $|\mathsf{aPKE}.\mathbb{R}^{\mathsf{Enc}}_{\mathsf{pp}}|$ and $|\mathsf{aPKE}.\mathbb{ST}^{\mathsf{aEnc}}_{\mathsf{pp}}|$ are polynomially lower bounded. We now prove correctness of RIdP, RIdPX, $\mathsf{RIdP}^\star$.

Theorem 29 (RIdP is Correct). *Let* PKE *be a correct public-key encryption scheme with injective randomness and* prF *be a pseudorandom function. Then* $\mathsf{aPKE} = \mathsf{RIdP}[\mathsf{PKE}, \mathsf{prF}]$ *(Construction 4) is correct. In particular, for all PPT adversaries $\mathcal{A}$ that make $q_E \leq |\mathsf{aPKE}.\mathbb{ST}^{\mathsf{aEnc}}_{\mathsf{pp}}|$ anamorphic-encrypt-then-anamorphic-decrypt queries, there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{B}}(\lambda) + q_E \frac{|\mathsf{aPKE}.\mathbb{AM}_{\mathsf{pp}}| - 1}{|\mathsf{aPKE}.\mathbb{R}^{\mathsf{Enc}}_{\mathsf{pp}}|}.$$

Proof. The proof follows identically to the proof of Theorem 14. Specifically, we can transition from a game $\mathcal{G}_0$ equivalent the correctness game $\mathcal{G}^{\mathsf{CORR}}$ on aPKE to a game $\mathcal{G}_1$ which replaces all prF outputs with true random samples. This transition is bounded by the PRF advantage of a reduction $\mathcal{B}$ which internally generates $(\mathsf{pk}, \mathsf{sk})$ and locally computes all encryption and decryption operations, using randomness from the prF oracle. Now because PKE is randomness injective, the adversary wins game $\mathcal{G}_1$ only in the event where the same r output for a queried amsg is also output by a different amsg′ on the same query. □

Theorem 30 (RIdPX is Correct). *Let* PKE *be a public-key encryption scheme with injective randomness and* prF *be a pseudorandom function. Then* $\mathsf{aPKE} = \mathsf{RIdPX}[\mathsf{PKE}, \mathsf{prF}]$ *(Construction 4) is correct. In particular, for all PPT adversaries* $\mathcal{A}$

$$\mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) = 0.$$

Proof. The proof follows identically to the proof of Theorem 19. Specifically, because the correctness game encrypts then decrypts once per query, and decryption always succeeds (r corresponding to amsg will decrypt if no r' corresponding to $\mathsf{amsg}' \neq \mathsf{amsg}$ does), then $\mathsf{ctr}_{\mathsf{aSign}} = \mathsf{ctr}_{\mathsf{aDec}}$ hence $\mathsf{prF}(\mathsf{ctr}_{\mathsf{aSign}}) = \mathsf{prF}(\mathsf{ctr}_{\mathsf{aDec}}) = r^*$ for each query. Then, since $\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}$ is a group, there cannot exist $\mathsf{amsg} \neq \mathsf{amsg}'$ such that $r^* \oplus \mathsf{amsg} = r^* \oplus \mathsf{amsg}'$. □

Theorem 31 (RIdP* is Correct). *Let* PKE *be a public-key encryption scheme with injective randomness and* prF *be a pseudorandom function. Then* $\mathsf{aPKE} = \mathsf{RIdP}^\star[\mathsf{PKE}, \mathsf{prF}]$ *(Construction 4) is correct. In particular, for all PPT adversaries* $\mathcal{A}$ *that make* $q_E \leq |\mathsf{aPKE}.\mathbb{ST}^{\mathsf{aEnc}}_{\mathsf{pp}}|$ *anamorphic-encrypt-then-anamorphic-decrypt queries, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{CORR}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) = \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{B}}(\lambda) + q_E \frac{|\mathsf{aPKE}.\mathbb{AM}_{\mathsf{pp}}| - 1}{|\mathsf{aPKE}.\mathbb{R}^{\mathsf{Enc}}_{\mathsf{pp}}|}.$$

Proof. The proof follows identically to the proof of Theorem 29, as the innocuous message msg is fixed each query and hence including it as input to the prF, which $\mathsf{RIdP}^\star$ does, has no effect. □

We now restate the stealthiness results on RIdP and RIdPX from BGHMR, and additionally briefly argue the stealthiness of $\mathsf{RIdP}^\star$.

Theorem 32 (Lemma 4.1 of [BGH+24]). *Let* PKE *be a public-key encryption scheme with injective randomness and* prF *be a pseudorandom function. Then* $\mathsf{aPKE} = \mathsf{RIdP}[\mathsf{PKE}, \mathsf{prF}]$ *(Construction 4) is stealthy. In particular, for all PPT adversaries* $\mathcal{A}$ *that make* $q_E \leq |\mathsf{aPKE}.\mathbb{ST}^{\mathsf{aEnc}}_{\mathsf{pp}}|$ *anamorphic encryption queries, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{B}}(\lambda).$$

Theorem 33 (Lemma 4.3 of [BGH+24]). *Let* PKE *be a public-key encryption scheme with injective randomness and* prF *be a pseudorandom function. Then* $\mathsf{aPKE} = \mathsf{RIdPX}[\mathsf{PKE}, \mathsf{prF}]$ *(Construction 4) is stealthy. In particular, for all PPT adversaries* $\mathcal{A}$ *that make* $q_E \leq |\mathsf{aPKE}.\mathbb{ST}^{\mathsf{aEnc}}_{\mathsf{pp}}|$ *anamorphic encryption queries, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{B}}(\lambda).$$

Theorem 34 (RIdP* is RoA-CAMA). *Let* PKE *be a public-key encryption scheme with injective randomness and* prF *be a pseudorandom function. Then* $\mathsf{aPKE} = \mathsf{RIdP}^\star[\mathsf{PKE}, \mathsf{prF}]$ *(Construction 4) is stealthy. In particular, for all PPT adversaries* $\mathcal{A}$ *that make* $q_E \leq |\mathsf{aPKE}.\mathbb{ST}^{\mathsf{aEnc}}_{\mathsf{pp}}|$ *anamorphic encrption queries, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{B}}(\lambda).$$

Proof. The proof follows identically to the proof of Theorem 32, except that the reduction $\mathcal{B}$'s prF oracle is queried with the counter, *honest message*, and anamorphic message as input. □

D.3 Robustness (for Anamorphic Encryption)

We restate BGHMR's robustness notion. Consider the $\mathcal{G}^{\mathsf{ROB\text{-}CMA}}$ game shown in Figure 27. The ROB-CMA advantage is defined by

$$\mathbf{Adv}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) = 2\Pr[\mathcal{G}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) \Rightarrow 1] - 1.$$

```
$\mathcal{G}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda)$
    $b \leftarrow\!\!\$ \{0,1\}$
    $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
    $(\mathsf{pk}, \mathsf{sk}, \mathsf{dk} :, \mathsf{st}_{\mathsf{aEnc}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow \mathsf{aPKE.aKeyGen}(\mathsf{pp})$
    $b^* \leftarrow \mathcal{A}^{O_{\mathsf{Sign},\mathsf{aDec}}}(\mathsf{pp})$
    $\textbf{return } [\![b = b^*]\!]$

$O_{\mathsf{Sign},\mathsf{aDec}}(\mathsf{msg}, \mathsf{st}^*_{\mathsf{aDec}})$
    $\textbf{if } b = 0 \textbf{ then return } \bot$
    $\mathsf{ct} \leftarrow \mathsf{aPKE.Enc}(\mathsf{pk}, \mathsf{msg})$
    $\mathsf{amsg} \leftarrow \mathsf{aPKE.aDec}(\mathsf{sk}, \mathsf{dk}, \mathsf{ct} : \mathsf{st}^*_{\mathsf{aDec}})$
    $\textbf{return } \mathsf{amsg}$
```

Fig. 27. Robustness game $\mathcal{G}^{\mathsf{ROB\text{-}CMA}}$ [BGH+24]

Definition 37. *An anamorphic encryption scheme* aPKE *is robust (in the* ROB-CMA *sense) if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda)$ *is negligible.*

With robustness now defined, we restate BGHMR's results on the robustness of RIdP and RIdPX. Since we will show that RIdP* satisfies dictator unforgeability, we do not analyze its robustness.

Theorem 35 (Lemma 4.2 of [BGH+24]). *Let* PKE *be a public-key encryption scheme with injective randomness and* prF *be a pseudorandom function. Then* aPKE = RIdP[PKE, prF] *and* aPKE = RIdPX[PKE, prF] *(Construction 4) are robust. In particular, for all PPT adversaries* $\mathcal{A}$ *that make* q_E *or fewer encrypt-then-anamorphic-decryption queries, there exists a PPT adversary such that*

$$\mathbf{Adv}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aPKE},\mathcal{A}}(\lambda) \leq q_E \frac{|\mathsf{aPKE}.\mathbb{AM}_{\mathsf{pp}}|}{|\mathsf{PKE}.\mathbb{R}^{\mathsf{Enc}}_{\mathsf{pp}}|}.$$

As discussed with Theorem 16, the Lemma in [BGH+24] included a PRF advantage term which is not actually needed.

D.4 Dictator Unforgeability (for Anamorphic Encryption)

The motivation behind dictator unforgeability presented in Section 4 applies identically to the anamorphic encryption setting since the dictator gets both public/verification and secret/signing key in either case. We now present the single and multi-challenge versions of dictator unforgeability for anamorphic encryption, which are both shown in Figure 28. We define the mc-DUF-CASA advantage by

$$\mathbf{Adv}^{\mathsf{mc\text{-}DUF\text{-}CACA}}_{\mathsf{aPKE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CACA}}_{\mathsf{aPKE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \Rightarrow 1].$$

By including msg in $\mathsf{L}_{\mathsf{aEnc}}$ and $\mathsf{L}_{\mathsf{aDec}}$, we explicitly require the anamorphic operations to authenticate the non-anamorphic message. For natural schemes, just including act in the lists would require the same anyway as act is anyway linked to msg via $\mathsf{msg} = \mathsf{aPKE.Dec}(\mathsf{sk}, \mathsf{act})$.

The following theorem relates the single and multi-challenge versions of dictator unforgeability for anamorphic encryption.

Theorem 36 (DUF-CASA ⇔ mc-DUF-CASA). *Let* aPKE *be an anamorphic encryption scheme. It is dictator unforgeable for triviality predicate* $\mathsf{pred}_{\mathsf{trivial}}$ *if and only if it is also multi-challenge dictator unforgeable for* $\mathsf{pred}_{\mathsf{trivial}}$*. In particular, for all PPT multi-challenge dictator unforgeability adversaries* $\mathcal{A}$ *that make* q_D *anamorphic decryption queries there exists a PPT dictator unforgeability adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{mc\text{-}DUF\text{-}CACA}}_{\mathsf{aPKE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \leq q_D \mathbf{Adv}^{\mathsf{DUF\text{-}CACA}}_{\mathsf{aPKE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{B}}(\lambda).$$

Furthermore, for all PPT dictator unforgeability adversaries $\mathcal{B}$*, there exists a PPT multi-challenge dictator unforgeability adversary* $\mathcal{A}$ *such that*

$$\mathbf{Adv}^{\mathsf{DUF\text{-}CACA}}_{\mathsf{aPKE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{B}}(\lambda) \leq \mathbf{Adv}^{\mathsf{mc\text{-}DUF\text{-}CACA}}_{\mathsf{aPKE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda).$$

```
$\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CACA}}_{\mathsf{aPKE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda)$
  $\mathsf{win} \leftarrow 0$
  $(\mathsf{L}_{\mathsf{aEnc}}, \mathsf{L}_{\mathsf{aDec}}) \leftarrow ([\cdot], [\cdot])$
  $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
  $(\mathsf{pk}, \mathsf{sk}, \mathsf{dk}, \mathsf{st}_{\mathsf{aEnc}}, \mathsf{st}_{\mathsf{aDec}}) \leftarrow \mathsf{aPKE.aKeyGen}(\mathsf{pp})$
  $\mathsf{act}^* \leftarrow \mathcal{A}^{O_{\mathsf{aEnc}}, O_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
  $\mathbf{return}\ \mathsf{win}$
  $\mathsf{amsg}^* \leftarrow \mathsf{aPKE.aDec}(\mathsf{sk}, \mathsf{dk}, \mathsf{act}^* : \mathsf{st}_{\mathsf{aDec}})$
  $\mathsf{msg}^* \leftarrow \mathsf{aPKE.Dec}(\mathsf{sk}, \mathsf{act})$
  $\mathsf{L}_{\mathsf{aDec}}.\mathsf{add}((\mathsf{msg}^*, \mathsf{act}^*))$
  $b_{\mathsf{valid}} \leftarrow [\![\mathsf{amsg}^* \neq \bot]\!]$
  $b_{\mathsf{trivial}} \leftarrow \mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{aEnc}}, \mathsf{L}_{\mathsf{aDec}})$
  $\mathbf{return}\ b_{\mathsf{valid}} \wedge \neg b_{\mathsf{trivial}}$

$O_{\mathsf{aEnc}}(\mathsf{msg}, \mathsf{amsg})$
  $\mathsf{act} \leftarrow \mathsf{aPKE.aEnc}(\mathsf{pk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{aEnc}})$
  $\mathsf{L}_{\mathsf{aEnc}}.\mathsf{add}((\mathsf{msg}, \mathsf{act}))$
  $\mathbf{return}\ \mathsf{act}$

$O_{\mathsf{aDec}}(\mathsf{msg}, \mathsf{act})$
  $\mathsf{msg} \leftarrow \mathsf{aPKE.Dec}(\mathsf{sk}, \mathsf{act})$
  $\mathsf{amsg} \leftarrow \mathsf{aPKE.aDec}(\mathsf{sk}, \mathsf{dk}, \mathsf{act} : \mathsf{st}_{\mathsf{aDec}})$
  $\mathsf{L}_{\mathsf{aDec}}.\mathsf{add}((\mathsf{msg}, \mathsf{act}))$
  $b_{\mathsf{valid}} \leftarrow [\![\mathsf{amsg} \neq \bot]\!]$
  $b_{\mathsf{trivial}} \leftarrow \mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{aEnc}}, \mathsf{L}_{\mathsf{aDec}})$
  $\mathbf{if}\ b_{\mathsf{valid}} \wedge \neg b_{\mathsf{trivial}}\ \mathbf{then}\ \mathsf{win} \leftarrow 1$
  $\mathbf{return}\ \mathsf{amsg}$
```

Fig. 28. Single- and multi-challenge dictator unforgeability game $\mathcal{G}^{\mathsf{DUF\text{-}CACA}}$ (without highlighted code) and $\mathcal{G}^{\mathsf{mc\text{-}DUF\text{-}CASA}}$ (including highlighted code)

Proof. The proof follows almost identically to the proof of Theorem 22, which addresses dictator unforgeability for anamorphic signature schemes. The second part of the theorem just defines $\mathcal{A}$ to forward all of $\mathcal{B}$'s queries to its own oracles and forward $\mathcal{B}$'s final output to its decryption query.

The first part constructs a $\mathcal{B}_i$ for $i \in \{1, \dots, q_D\}$ which forwards $\mathcal{A}$'s queries to its own oracles until it outputs $\mathcal{A}$'s i^{th} decryption query as a forgery attempt. The overall bound follows by having $\mathcal{B}$ run a $\mathcal{B}_i$ chosen at random. □

With this bound established, we can now show the anamorphic encryption equivalent of the result that schemes that are indistinguishable under chosen anamorphic message attack, when dictator unforgeable, are also indistinguishable under chosen anamorphic *ciphertext* attack.

Theorem 37 (IND-CAMA + DUF-CACA ⇒ IND-CACA). *Let* aPKE *be an* IND-CAMA *indistinguishable and dictator unforgeable anamorphic encryption scheme for triviality* $\mathsf{pred}_{\mathsf{trivial}}$*. Then it is also* IND-CACA *indistinguishable for* $\mathsf{pred}_{\mathsf{trivial}}$*. In particular, for all PPT adversaries* $\mathcal{A}$ *that make at most* q_D *anamorphic decryption queries there exist PPT adversaries* $\mathcal{B}_0$ *and* $\mathcal{B}_1$ *such that*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CACA}}_{\mathsf{aPKE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \le 2q_D\, \mathbf{Adv}^{\mathsf{DUF\text{-}CACA}}_{\mathsf{aPKE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{B}_0}(\lambda) + \mathbf{Adv}^{\mathsf{IND\text{-}CAMA}}_{\mathsf{aPKE},\mathcal{B}_1}(\lambda).$$

Proof. The proof follows almost identically to the proof of Theorem 2, which addresses confidentiality and dictator unforgeability for anamorphic signature schemes. The proof proceeds by first hopping from a game $\mathcal{G}_0$ equivalent to $\mathcal{A}$ playing the $\mathcal{G}^{\mathsf{IND\text{-}CACA}}$ game on aPKE to a game $\mathcal{G}_1$ which responds to all anamorphic decryption queries with $\bot$. This transition is bounded by the mc-DUF-CACA advantage of a PPT reduction $\mathcal{B}_0'$ which itself is bounded by the DUF-CACA advantage of a PPT $\mathcal{B}_0$ per Theorem 36. Since the final game $\mathcal{G}_1$ never answers any anamorphic decryption queries, defeating this game is equivalent to breaking IND-CAMA security, i.e., given $\mathcal{A}$ we can construct a PPT $\mathcal{B}_1$ that simulates $\mathcal{G}_1$ to $\mathcal{A}$ by returning $\bot$ to all decryption queries and uses it to win the $\mathcal{G}^{\mathsf{IND\text{-}CAMA}}$ game. □

D.5 Dictator Forgeability of (Anamorphic Encryption Versions of) RRep, RIdP, and RIdPX

We now provide anamorphic encryption analogues of our dictator forgery attacks presented in Section 4. Schemes RRep, RIdP and RIdPX are insecure due to "re-encrypting" attacks. The first two can be easily patched (to RRep* and RIdP*, specified in Section 3) while the third has no natural fix.

Dictator Forgeability of RRep. Consider $\mathsf{RRep}[\mathsf{PKE}, \mathsf{prE}]$, for which anamorphic ciphertexts are computed via $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}; r)$ where $r \leftarrow \mathsf{prE.Enc}(\mathsf{dk}, \mathsf{amsg})$.

The dictator can attack this scheme as follows. Suppose a party under the dictator's domain sends an act that contains a valid anamorphic message amsg and encrypts an innocuous message msg. The dictator, who knows the secret key, can then extract $r \leftarrow \mathsf{PKE.RRecov}(\mathsf{pk}, \mathsf{sk}, \mathsf{act})$ and produce a *new* anamorphic ciphertext by selecting a fresh innocuous msg^* and encrypting $\mathsf{act}^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}^*; r)$. The dictator sends along act^* as their forgery, and a recipient who runs RRep's anamorphic decryption procedure first obtains $r \leftarrow \mathsf{PKE.RRecov}(\mathsf{pk}, \mathsf{sk}, \mathsf{act}^*)$ before computing $\mathsf{amsg}^* \leftarrow \mathsf{prE.Dec}(\mathsf{dk}, r^*)$. Observe that $r^* = r$, so $\mathsf{amsg}^* = \mathsf{amsg} \neq \perp$; hence, act^* is a valid forgery.

Dictator Forgeability of RIdP. Consider $\mathsf{RIdP}[\mathsf{PKE}, \mathsf{prF}]$, for which $\mathsf{act} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}; r)$ for $r \leftarrow \mathsf{prF}(\mathsf{dk}, (\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{amsg}))$. Furthermore, suppose PKE is randomness recovering.

The dictator can attack this scheme as follows. Given an anamorphic ciphertext act that encrypts msg and hides a amsg, the dictator can extract $r \leftarrow \mathsf{PKE.RRecov}(\mathsf{pk}, \mathsf{sk}, \mathsf{act})$ and compute $\mathsf{act}^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}^*; r)$ for a new msg^*. Since decryption computes $r' \leftarrow \mathsf{prF}(\mathsf{dk}, (\mathsf{ctr}_{\mathsf{aEnc}}, \mathsf{amsg}'))$ for all possible amsg' and outputs an r' used to generate act^*, then anamorphic decryption outputs $\mathsf{amsg} \neq \perp$ and act^* is a valid forgery.

Dictator Forgeability of RIdPX. Consider $\mathsf{RIdPX}[\mathsf{PKE}, \mathsf{prF}]$, for which anamorphic ciphertexts are computed via $\mathsf{act} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}; r)$ where $r \leftarrow \mathsf{prF}(\mathsf{dk}, \mathsf{ctr}_{\mathsf{aEnc}}) \oplus \mathsf{amsg}$. Now suppose that PKE is the ElGamal encryption scheme which, for a keypair $(\mathsf{pk}, \mathsf{sk}) = (g^x, x)$, has ciphertexts of the form $(g^r, \mathsf{pk}^r \cdot \mathsf{msg})$. For this, $\mathsf{PKE}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}} = \mathbb{Z}_p$ and $\oplus$ denotes addition modulo p. Note that BGHMR use ElGamal as an example instantiation for RIdPX.

A dictator can perform the following attack on this scheme. Let $\mathsf{act} = (c_1, c_2) = (g^r, \mathsf{pk}^r \cdot \mathsf{msg})$ where $r \leftarrow \mathsf{prF}(\mathsf{k}, \mathsf{ctr}_{\mathsf{aEnc}}) \oplus \mathsf{amsg}$. The dictator can simply compute $\mathsf{act}^* = (c_1 \cdot g^{r'}, c_2 \cdot g^{r'})$, which they submit as their forgery. If r' is chosen such that $\mathsf{amsg} \oplus r' \in \mathsf{aPKE}_{\mathsf{RIdPX\text{-}DF}}.\mathbb{AM}_{\mathsf{pp}}$, then act^* decrypts to a *new* valid anamorphic message and not $\perp$. Similar to the signature scheme version, the dictator may be able to forge a new anamorphic signature that encrypts *any* valid anamorphic message in the anamorphic message space. Unlike the signature scheme attack, however, the secret key was never required to create the forgery, so *anyone* (dictator or even regular user) can create a dictator forgery.

D.6 Dictator Unforgeability of $\mathsf{RRep}^\star$ and $\mathsf{RIdP}^\star$

We now show that the anamorphic encryption versions of our improved $\mathsf{RRep}^\star$ and $\mathsf{RIdP}^\star$ transforms are dictator unforgeable.

Dictator Unforgeability of $\mathsf{RRep}^\star$. The proof for anamorphic encryption $\mathsf{RRep}^\star$ is similar to the proof for the anamorphic signature version, with some minor differences. Firstly, the anamorphic signature version relied on a *strong* randomness-recovery property which is not necessary for the anamorphic encryption version because it performs a re-encryption check.

Theorem 38 ($\mathsf{RRep}^\star$ is $\mathsf{DUF\text{-}CACA}$). *Let* PKE *be a public-key encryption scheme and* prE *be an AEAD scheme that achieves ciphertext integrity for equality-pattern-respecting triviality predicate* $\mathsf{pred}_{\mathsf{trivial}}$ *(which is permissive if* prE *is). Then* $\mathsf{aPKE} = \mathsf{RRep}^\star[\mathsf{PKE}, \mathsf{prE}]$ *(Construction 3) is dictator unforgeable for* $\mathsf{pred}_{\mathsf{trivial}}$*. In particular, for all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{DUF\text{-}CACA}}_{\mathsf{aPKE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{INT\text{-}CT}}_{\mathsf{prE},\mathsf{pred}_{\mathsf{trivial}},\mathcal{B}}(\lambda).$$

Proof. The proof follows that of Theorem 3. We start with a minor rewriting step where we stop adding $(\mathsf{msg}, \mathsf{act})$ to $\mathsf{L}_{\mathsf{aDec}}$ if decrypting act failed. Then the main step hops to a game which computes the triviality predicate using the inputs and outputs of prE in lists rather than aPKE.

Consider the function f defined by $f((\mathsf{msg}, \mathsf{act})) = (\mathsf{msg}, r)$ when $r = \mathsf{PKE.RRecov}(\mathsf{pk}, \mathsf{sk}, \mathsf{act}) \neq \perp$ and $\mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}; r) = \mathsf{act}$. Otherwise $f((\mathsf{msg}, \mathsf{asig})) = \perp$. This is quasi-injective. Specifically, if

$f((\mathsf{msg}, \mathsf{act})) = f((\mathsf{msg}', \mathsf{act}'))$, then clearly $\mathsf{msg} = \mathsf{msg}'$. But then $\mathsf{act} = \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}; r) = \mathsf{act}'$. The lists in the new game will be $f(\mathsf{L}_{\mathsf{aEnc}}), f(\mathsf{L}_{\mathsf{aDec}})$. Equality-pattern respecting gives that

$$\mathsf{pred}_{\mathsf{trivial}}(\mathsf{L}_{\mathsf{aEnc}}, \mathsf{L}_{\mathsf{aDec}}) = \mathsf{pred}_{\mathsf{trivial}}(f(\mathsf{L}_{\mathsf{aEnc}}), f(\mathsf{L}_{\mathsf{aDec}})).$$

A simple reduction $\mathcal{B}$ shows winning this game is equivalent to breaking INT-CT security of prE. □

Dictator Unforgeability of RIdP⋆. The proof for anamorphic encryption RIdP⋆ is similar to the proof for the anamorphic signature version, with the minor difference being that randomness-injectivity is sufficient, unlike the signature scheme version which required a strong randomness identification procedure.

Theorem 39 (RIdP⋆ is mc-DUF-CACA). *Let* PKE *be a randomness-injective public-key encryption scheme and* prF *be pseudorandom function. Then* $\mathsf{aPKE} = \mathsf{RIdP}^{\star}[\mathsf{PKE}, \mathsf{prF}]$ *(Construction 4) is multi-challenge dictator unforgeable dictator unforgeable for* $\mathsf{pred}^{\mathsf{psynch}}_{\mathsf{trivial}}$. *In particular, for all PPT adversaries* $\mathcal{A}$ *that make at most* q_D *anamorphic decryption queries and at most* $|\mathsf{aPKE}.\mathbb{ST}^{\mathsf{aEnc}}_{\mathsf{pp}}|$ *queries to either oracle, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{mc\text{-}DUF\text{-}CACA}}_{\mathsf{aPKE},\mathsf{pred}^{\mathsf{psynch}}_{\mathsf{trivial}},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{B}}(\lambda) + \frac{q_D|\mathsf{aPKE}.\mathbb{AM}_{\mathsf{pp}}|}{|\mathsf{aPKE}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}|}.$$

Proof. The proof follows that of Theorem 23. A PRF reduction switches the prF to a random function. Then we argue that the ctr computed inside $\mathsf{pred}^{\mathsf{psynch}}_{\mathsf{trivial}}$ matches that anamorphic decryption counter $\mathsf{ctr}_{\mathsf{aDec}}$. Then forging requires guessing the output of the random function on a $(\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{msg}, \mathsf{amsg})$ other than the one used in anamorphic decryption which has an at most $|\mathsf{aPKE}.\mathbb{AM}_{\mathsf{pp}}|/|\mathsf{aPKE}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}|$ probability of occurring with each anamorphic decryption query. □

E Details on Strengthening Private Anamorphism to Recipient Unforgeability

E.1 Full Proof of Theorem 7 (RRep⋆ with SUF-CRA Signatures is RUF-CAMA)

In Section 5, we argued that RRep⋆, when instantiated with chosen randomness unforgeable signature schemes, produce recipient unforgeable anamorphic signature schemes. We provide a full proof here.

Theorem 7 (RRep⋆ with SUF-CRA Signatures is RUF-CAMA). *Let* S *be a chosen-randomness unforgeable publicly randomness-recovering signature scheme and* prE *be a pseudorandom encryption scheme. Then* $\mathsf{aS} = \mathsf{RRep}^{\star}[\mathsf{S}, \mathsf{prE}]$ *(Construction 1) is recipient unforgeable. In particular, for all PPT adversaries* $\mathcal{A}$*, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{S},\mathcal{B}}(\lambda).$$

Proof. Let $\mathcal{A}$ be a PPT recipient forger for aS. We will use $\mathcal{A}$ to construct a PPT adversary $\mathcal{B}$ that breaks the SUF-CRA-security of S. We first review the games that $\mathcal{A}$ and $\mathcal{B}$ play. $\mathcal{A}$ plays the $\mathcal{G}^{\mathsf{RUF\text{-}CAMA}}$ game, where it receives a verification key vk, double key dk, and initial anamorphic decryption state $\mathsf{st}_{\mathsf{Dec}}$, and has access to a signing oracle $O^{\mathcal{A}}_{\mathsf{Sign}}$ and an anamorphic signing oracle $O^{\mathcal{A}}_{\mathsf{aSign}}$. $\mathcal{B}$ plays the $\mathcal{G}^{\mathsf{SUF\text{-}CRA}}$ game, where it receives a verification key vk and has access to a signing oracle $O^{\mathcal{B}}_{\mathsf{Sign}}$, to which it can query for signatures on randomness of its choosing.

We construct $\mathcal{B}$ from $\mathcal{A}$ as shown in Figure 29. Upon initialization, $\mathcal{B}$ generates $(\mathsf{dk}, \mathsf{st}_{\mathsf{Enc}}, \mathsf{st}_{\mathsf{Dec}}) \leftarrow \mathsf{prE.KeyGen}(\mathsf{pp})$ and gives vk, dk, and $\mathsf{st}_{\mathsf{Dec}}$ to $\mathcal{A}$. When $\mathcal{A}$ queries $O^{\mathcal{A}}_{\mathsf{Sign}}$ with msg, $\mathcal{B}$ samples some randomness r and queries its signing oracle $O^{\mathcal{B}}_{\mathsf{Sign}}$ on (msg, r). When $\mathcal{A}$ queries $O^{\mathcal{A}}_{\mathsf{aSign}}$ with $(\mathsf{msg}, \mathsf{amsg})$ for an anamorphic signature, $\mathcal{B}$ computes the pseudorandom encryption of amsg with associated data msg to obtain a ciphertext act (i.e. it computes $\mathsf{act} \leftarrow \mathsf{prE.Enc}(\mathsf{k}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{Enc}})$) and queries $(\mathsf{msg}, \mathsf{act})$ to its signing oracle $O^{\mathcal{B}}_{\mathsf{Sign}}$. Once $\mathcal{A}$ outputs a forgery $(\mathsf{msg}^*, \mathsf{sig}^*)$, $\mathcal{B}$ forwards this as its forgery. Note that $\mathcal{B}$'s forgery will be new if $\mathcal{A}$'s is, since it directly forwarded $\mathcal{A}$'s signing queries. Furthermore, from $\mathcal{A}$'s perspective, it has exactly played the recipient unforgeability game $\mathcal{G}^{\mathsf{RUF\text{-}CAMA}}$ on aS, hence $\mathcal{B}$'s advantage is identical to $\mathcal{A}$'s advantage. Finally, $\mathcal{B}$ is PPT because $\mathcal{A}$ prE.KeyGen, and prE.Enc are PPT. The theorem bound follows from these observations. □

```
$\mathcal{B}^{O^{\mathcal{B}}_{\mathsf{Sign}}}(\mathsf{pp}, \mathsf{vk})$
    $(\mathsf{dk}, \mathsf{st}_{\mathsf{Enc}}, \mathsf{st}_{\mathsf{Dec}}) \leftarrow \mathsf{prE.KeyGen}()$
    $(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}^{O^{\mathcal{A}}_{\mathsf{Sign}}, O^{\mathcal{A}}_{\mathsf{aSign}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{dk}, \mathsf{st}_{\mathsf{Dec}})$
    $\textbf{return } (\mathsf{msg}^*, \mathsf{sig}^*)$
```

```
$O^{\mathcal{A}}_{\mathsf{Sign}}(\mathsf{msg})$
    $r \leftarrow\!\!\$\ \mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}$
    $\mathsf{sig} \leftarrow O^{\mathcal{B}}_{\mathsf{Sign}}(\mathsf{msg}, r)$
    $\textbf{return } \mathsf{sig}$
```

```
$O^{\mathcal{A}}_{\mathsf{aSign}}(\mathsf{msg}, \mathsf{amsg})$
    $\mathsf{act} \leftarrow \mathsf{prE.Enc}(\mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{Enc}})$
    $\mathsf{asig} \leftarrow O^{\mathcal{B}}_{\mathsf{Sign}}(\mathsf{msg}, \mathsf{act})$
    $\textbf{return } \mathsf{asig}$
```

Fig. 29. Reduction used in the proof of Theorem 7

E.2 Full Proof of Corollary 4 (RRep*[RSA-PSS, prE] is RUF-CAMA)

In Section 5, we argued that RSA-PSS signatures are existentially unforgeable under chosen randomness attack. We provide a full proof here.

Before continuing, we introduce a modified RSA* assumption captured by the game in Figure 30, which also shows a game for the standard RSA assumption. The advantage is defined straightforwardly for each. We say that an adversary is admissible if it makes up to q_C challenge queries and it never queries on any $i > q_C$. Note that the modified assumption is not only multi-challenge with adaptive corruptions, but also samples challennges bitstrings of length $\lambda - 1$ rather than uniformly from $\mathbb{Z}_N$ as is done in the standard assumption. This assumption simplifies the subsequent proof and is bounded by the standard RSA assumption, as shown in the following lemma.

Lemma 2. *The multi-challenge RSA* assumption under adaptive corruption is hard if the standard RSA assumption is hard. In particular, for all admissible PPT adversaries $\mathcal{A}$ that make q_C challenge queries, there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{mc\text{-}RSA^*\text{-}ac}}_{\mathcal{A}}(\lambda) \leq 2q_C\, \mathbf{Adv}^{\mathsf{RSA}}_{\mathcal{B}}(\lambda) + q_C^2 2^{-(\lambda-1)}.$$

Proof. The proof follows from a series of game hops from game $\mathcal{G}_0$, equivalent to an adversary $\mathcal{A}$ playing the $\mathcal{G}^{\mathsf{mc\text{-}RSA^*\text{-}ac}}$ game, to game $\mathcal{G}_4$, where $\mathcal{A}$'s probability of winning is bounded by a PPT reduction $\mathcal{B}$ against the $\mathcal{G}^{\mathsf{RSA}}$ game. The games are shown in Figure 30.

Game $\mathcal{G}_0$. This game is equivalent to the $\mathcal{G}^{\mathsf{mc\text{-}RSA^*\text{-}ac}}$ game, except that we have moved the code in the O_{Chal} oracle that samples fresh y into its own sampling procedure $\mathsf{S1}$. Thus,

$$\mathbf{Adv}^{\mathsf{mc\text{-}RSA^*\text{-}ac}}_{\mathcal{A}}(\lambda) = \Pr[\mathcal{G}_0(\lambda) \Rightarrow 1].$$

Transition from game $\mathcal{G}_0$ to $\mathcal{G}_1$. This transition jumps to a game which randomly guesses an index i' of the challenge query i^* for which $\mathcal{A}$ will output a pre-image x^*. Because $\mathcal{A}$ is admissible and makes q_C or fewer queries and i' is sampled from $\{1, \ldots, q_C\}$, the probability $i' = i^*$ is $1/q_C$. If the game guesses incorrectly, it outputs 0. Since the query guess i' is independent of the adversary's view,

$$\Pr[\mathcal{G}_0(\lambda) \Rightarrow 1] = q_C \Pr[\mathcal{G}_1(\lambda) \Rightarrow 1].$$

This transition also makes a minor cosmetic change in how the O_{Chal} oracle is written. Specifically, on new i, the oracle checks whether or not $i = i'$ but still samples y via $\mathsf{S1}$ in either case.

Transition from game $\mathcal{G}_1$ to $\mathcal{G}_2$. This transition modifies the way y is sampled when the O_{Chal} oracle is queried on $i = i'$ for the first time. Instead of using $\mathsf{S1}$, which samples $\tilde{y} \leftarrow\!\!\$\ \{0,1\}^{\lambda-1}$ and sets $y \leftarrow 0\|\tilde{y}$, it uses $\mathsf{S2}$, which simply samples $y \leftarrow\!\!\$\ \mathbb{Z}_N$. These two games are identical, except if y does not start with a 0 bit, which must occur with probability at most $1/2$. It follows that

$$\Pr[\mathcal{G}_1(\lambda) \Rightarrow 1] \leq 2 \Pr[\mathcal{G}_2(\lambda) \Rightarrow 1].$$

```
$\mathcal{G}^{\mathsf{RSA}}_{\mathcal{A}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^{\lambda})$
$p \neq q \leftarrow\!\!\$\ \mathbb{P}_{\lambda/2}$
$N \leftarrow pq$
$e \leftarrow \mathsf{RSA.eGen}(\mathsf{pp}, \varphi(N))$
$d \leftarrow e^{-1} \pmod{\varphi(N)}$
$y^* \leftarrow\!\!\$\ \mathbb{Z}_N$
$x^* \leftarrow \mathcal{A}(\mathsf{pp}, N, e, y^*)$
return $[\![y^* = x^{*e} \pmod{N}]\!]$
```

```
$\mathcal{G}^{\mathsf{mc\text{-}RSA^*\text{-}ac}}_{\mathcal{A}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^{\lambda})$
$p \neq q \leftarrow\!\!\$\ \mathbb{P}_{\lambda/2}$
$N \leftarrow pq$
$e \leftarrow \mathsf{RSA.eGen}(\mathsf{pp}, \varphi(N))$
$d \leftarrow e^{-1} \pmod{\varphi(N)}$
$(x^*, i^*) \leftarrow \mathcal{A}^{O_{\mathsf{Chal}}, O_{\mathsf{Corr}}}(\mathsf{pp}, N, e)$
$b_{\mathsf{valid}} \leftarrow [\![\mathsf{T_y}[i^*] = x^{*e} \pmod{N}]\!]$
$b_{\mathsf{trivial}} \leftarrow [\![\mathsf{T_x}[i^*] \neq \perp]\!]$
return $b_{\mathsf{valid}} \wedge \neg b_{\mathsf{trivial}}$
```

```
$O_{\mathsf{Chal}}(i)$
if $\mathsf{T_y}[i] = \perp$ then
    $\tilde{y} \leftarrow\!\!\$\ \{0,1\}^{\lambda-1}$
    $\mathsf{T_y}[i] \leftarrow 0 \| \tilde{y}$
return $\mathsf{T_y}[i]$
```

```
$O_{\mathsf{Corr}}(i)$
$y \leftarrow \mathsf{T_y}[i]$
$\mathsf{T_x}[i] \leftarrow y^d \pmod{N}$
return $\mathsf{T_x}[i]$
```

```
$\mathcal{G}_n$ for $n \in \{0, 1, 2, 3, 4\}$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^{\lambda})$
$i' \leftarrow\!\!\$\ \{1, \ldots, q_C\}$  // $\mathcal{G}_{[1,\infty)}$
$p \neq q \leftarrow\!\!\$\ \mathbb{P}_{\lambda/2}$
$N \leftarrow pq$
$e \leftarrow \mathsf{RSA.eGen}(\mathsf{pp}, \varphi(N))$
$d \leftarrow e^{-1} \pmod{\varphi(N)}$
$(x^*, i^*) \leftarrow \mathcal{A}^{O_{\mathsf{Chal}}, O_{\mathsf{Corr}}}(\mathsf{pp}, N, e)$
$b_{\mathsf{valid}} \leftarrow [\![\mathsf{T_y}[i^*] = x^{*e} \pmod{N}]\!]$
$b_{\mathsf{trivial}} \leftarrow [\![\mathsf{T_x}[i^*] \neq \perp]\!]$
$b_{\mathsf{guess}} \leftarrow [\![i' = i^*]\!]$  // $\mathcal{G}_{[1,\infty)}$
return $b_{\mathsf{valid}} \wedge \neg b_{\mathsf{trivial}}$
return $b_{\mathsf{valid}} \wedge \neg b_{\mathsf{trivial}} \wedge b_{\mathsf{guess}}$  // $\mathcal{G}_{[1,\infty)}$
```

```
$O_{\mathsf{S3}}()$
$\mathsf{ctr} \leftarrow 1$  // $\mathcal{G}_{[4,\infty)}$
while $y \neq 0 \| \tilde{y}$ do  // $\mathcal{G}_{[3,4)}$
while $y \neq 0 \| \tilde{y} \wedge \mathsf{ctr} \leq \lambda$ do  // $\mathcal{G}_{[4,\infty)}$
    $x \leftarrow\!\!\$\ \mathbb{Z}_N$
    $y \leftarrow x^e \pmod{N}$
    $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$  // $\mathcal{G}_{[4,\infty)}$
return $(x, y)$
```

```
$O_{\mathsf{Chal}}(i)$
if $\mathsf{T_y}[i] = \perp$ then
    $y \leftarrow \mathsf{S1}()$  // $\mathcal{G}_{[0,\infty)}$
    if $i = i'$ then  // $\mathcal{G}_{[1,\infty)}$
        $y \leftarrow \mathsf{S1}()$  // $\mathcal{G}_{[1,2)}$
        $y \leftarrow \mathsf{S2}()$  // $\mathcal{G}_{[2,\infty)}$
    else  // $\mathcal{G}_{[1,\infty)}$
        $y \leftarrow \mathsf{S1}()$  // $\mathcal{G}_{[1,3)}$
        $(x, y) \leftarrow \mathsf{S3}()$  // $\mathcal{G}_{[3,\infty)}$
        $\mathsf{T_x}[i] \leftarrow x$  // $\mathcal{G}_{[3,\infty)}$
    $\mathsf{T_y}[i] \leftarrow y$
return $\mathsf{T_y}[i]$
```

```
$O_{\mathsf{Corr}}(i)$
$y \leftarrow \mathsf{T_y}[i]$  // $\mathcal{G}_{[0,3)}$
$\mathsf{T_x}[i] \leftarrow y^d \pmod{N}$  // $\mathcal{G}_{[0,3)}$
return $\mathsf{T_x}[i]$
```

```
$O_{\mathsf{S1}}()$
$\tilde{y} \leftarrow\!\!\$\ \{0,1\}^{\lambda-1}$
return $0 \| \tilde{y}$
```

```
$O_{\mathsf{S2}}()$
$y \leftarrow\!\!\$\ \mathbb{Z}_N$
return $y$
```

```
$\mathcal{B}(\mathsf{pp}, N, e, y^*)$
$i' \leftarrow\!\!\$\ \{1, \ldots, q_C\}$
$(x^*, \cdot) \leftarrow \mathcal{A}^{O_{\mathsf{Chal}}, O_{\mathsf{Corr}}}(\mathsf{pp}, N, e)$
return $x^*$
```

```
$O_{\mathsf{Chal}}(i)$
if $\mathsf{T_y}[i] = \perp$ then
    if $i = i'$ then
        $y \leftarrow y^*$
    else
        $(x, y) \leftarrow \mathsf{S3}()$
        $\mathsf{T_x}[i] \leftarrow x$
    $\mathsf{T_y}[i] \leftarrow y$
return $\mathsf{T_y}[i]$
```

```
$O_{\mathsf{Corr}}(i)$
return $\mathsf{T_x}[i]$
```

```
$O_{\mathsf{S3}}()$
$\mathsf{ctr} \leftarrow 1$
while $y \neq 0 \| \tilde{y} \wedge \mathsf{ctr} \leq \lambda$ do
    $x \leftarrow\!\!\$\ \mathbb{Z}_N$
    $y \leftarrow x^e \pmod{N}$
    $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
return $(x, y)$
```

Fig. 30. Standard RSA game $\mathcal{G}^{\mathsf{RSA}}$ (top left) and modified RSA game $\mathcal{G}^{\mathsf{mc\text{-}RSA^*\text{-}ac}}$ (top right), and games (bottom left) and reduction (bottom right) used in the proof of Lemma 2

Transition from game $\mathcal{G}_2$ to $\mathcal{G}_3$. This transition modifies the way y is sampled when the O_{Chal} oracle is queried on any $i \neq i'$ for the first time. Instead of using $\mathsf{S1}$, it uses $\mathsf{S3}$, which rejection samples $x \leftarrow\!\!\$\ \mathbb{Z}_N$ until $y \leftarrow x^e \pmod{N}$ where y starts with a 0 bit. The sampled x is stored and output for any appropriate corruption query. The distribution of (x, y) is identical in both games because RSA is a permutation. The games are thus identical, except when the adversary queries i' to the O_{Corr} oracle. In this event, the adversary loses by definition, so

$$\Pr[\mathcal{G}_2(\lambda) \Rightarrow 1] = \Pr[\mathcal{G}_3(\lambda) \Rightarrow 1].$$

Transition from game $\mathcal{G}_3$ to $\mathcal{G}_4$. This transition modifies the rejection sampling code performed in S3 so that it only runs up to λ iterations instead of indefinitely. This game and the previous are identical-until-bad, where the bad event occurs if all λ iterations of rejection sampling fail during any challenge query. Since x is uniform and RSA is a permutation, the probability y doesn't start with a 0 bit is at most 1/2 for each iteration. This event can occur over any of the q_C challenge queries, so

$$\Pr[\mathcal{G}_3(\lambda) \Rightarrow 1] - \Pr[\mathcal{G}_4(\lambda) \Rightarrow 1] \leq q_C 2^{-\lambda}.$$

Alternatively, we could have bounded the game to globally perform at most $2q_C$ iterations of this loop for a multiplicative bound $\Pr[\mathcal{G}_3(\lambda) \Rightarrow 1] \leq 2\Pr[\mathcal{G}_4(\lambda) \Rightarrow 1]$ since at least half the time, half of the iterations of the loop will suceed.

Game $\mathcal{G}_4$. This game is equivalent to reduction $\mathcal{B}$ shown in Figure 30 playing the standard RSA game $\mathcal{G}^{\mathsf{RSA}}$ game. It follows that

$$\Pr[\mathcal{G}_4(\lambda) \Rightarrow 1] = \mathbf{Adv}^{\mathsf{RSA}}_{\mathcal{B}}(\lambda).$$

The theorem bound follows from these observations. □

Lemma 1 (RSA-PSS is SUF-CRA). *RSA-PSS is chosen-randomness unforgeable, provided the hash functions H and G underlying it are modeled as random oracles and λ_0 is super-logarithmic in λ. In particular, for all PPT adversaries $\mathcal{A}$ that make up to q_S signing queries and q_H and q_G hash queries (to hash functions H and G respectively), there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{RSA\text{-}PSS},\mathcal{A}}(\lambda) \leq (q_S + q_H + 1)\left(2\,\mathbf{Adv}^{\mathsf{RSA}}_{\mathcal{B}}(\lambda) + (q_S + q_H + 1)2^{-\lambda-1} + (q_S + q_H + q_G + 1)2^{-\lambda_0}\right).$$

Proof. The proof follows from a series of game hops from game $\mathcal{G}_0$, equivalent to an adversary $\mathcal{A}$ playing the $\mathcal{G}^{\mathsf{SUF\text{-}CRA}}$ game, to game $\mathcal{G}_8$, where $\mathcal{A}$'s probability of winning is bounded by a PPT reduction $\mathcal{B}'$ against the $\mathcal{G}^{\mathsf{mc\text{-}RSA^*\text{-}ac}}$ game, which we bound by the hardness of standard RSA by applying Lemma 2. The games are shown in Figure 31. Because they are have similar code, we write O_{Sign} and O_H overlaid. We let $\lambda_2 = \lambda - \lambda_0 - \lambda_1 - 1$.

Game $\mathcal{G}_0$. This game is equivalent to $\mathcal{G}^{\mathsf{SUF\text{-}CRA}}$ with lazy-sampled random oracles, with two additional modifications. Firstly, the table underlying H, instead of being indexed by $(\mathsf{sig}, \mathsf{msg})$, uses incrementing indices i the game maintains via a stateful function M. Secondly, the check that a forgery is new uses a table $\mathsf{T}_{\mathsf{Sign}}$ indexed by i instead of a set S. This is identical because both M and the mapping between (msg, r) and $(\mathsf{msg}, \mathsf{sig})$ for valid $(\mathsf{msg}, \mathsf{sig})$ under vk are bijections. The latter follows from the fact RSA-PSS is strongly publicly randomness recovering. We have that,

$$\mathbf{Adv}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{RSA\text{-}PSS},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}_0(\lambda) \Rightarrow 1].$$

Transition from game $\mathcal{G}_0$ to $\mathcal{G}_1$. This transition adds the win condition $\alpha^* = O_{G_1}(w^*) \oplus r^*$, which holds trivially given that r^* is defined as $O_{G_1}(w^*) \oplus \alpha^*$, making the games identical. Hence

$$\Pr[\mathcal{G}_0(\lambda) \Rightarrow 1] = \Pr[\mathcal{G}_1(\lambda) \Rightarrow 1].$$

```
$\mathcal{G}_i$ for $i \in \{0, 1, \ldots, 8\}$
$\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
$\mathsf{ctr} \leftarrow 1$
$p \neq q \leftarrow\$ \mathbb{P}_{\lambda/2}$
$N \leftarrow pq$
$e \leftarrow \mathsf{RSA.eGen}(\mathsf{pp}, \varphi(N))$
$d \leftarrow e^{-1} \pmod{\varphi(N)}$
$(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}^{O_{\mathsf{Sign}}}(\mathsf{pp}, (N, e))$
$(b^* \| w^* \| \alpha^* \| \gamma^*) \leftarrow (\mathsf{sig}^*)^e \pmod N$
$r^* \leftarrow O_{G_1}(w^*) \oplus \alpha^*$
$i^* \leftarrow \mathsf{M}[\mathsf{msg}^*, r^*]$
$b_{\mathsf{valid}} \leftarrow [\![b^* = 0]\!]$
              $\wedge [\![w^* = O_H(\mathsf{msg}^*, r^*)]\!]$
              $\wedge [\![\alpha^* = O_{G_1}(w^*) \oplus r^*]\!]$ // $\mathcal{G}_{[1,\infty)}$
              $\wedge [\![\gamma^* = O_{G_2}(w^*)]\!]$
$b_{\mathsf{new}} \leftarrow [\![\mathsf{T}_{\mathsf{Sign}}[i^*] \neq \bot]\!]$
return $b_{\mathsf{valid}} \wedge b_{\mathsf{new}}$
```

```
$\mathsf{M}(\mathsf{msg}, r)$
if $\mathsf{T}_\mathsf{M}[\mathsf{msg}, r] = \bot$ then
    $\mathsf{T}_\mathsf{M}[\mathsf{msg}, r] \leftarrow \mathsf{ctr}$
    $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
return $\mathsf{T}_\mathsf{M}[\mathsf{msg}, r]$
```

```
$O_{G_1}(w)$
if $\mathsf{T}_{G_1}[w] = \bot$ then
    $\mathsf{T}_{G_1}[w] \leftarrow\$ \{0,1\}^{\lambda_1}$
return $\mathsf{T}_{G_1}[w]$
```

```
$O_{G_2}(w)$
if $\mathsf{T}_{G_2}[w] = \bot$ then
    $\mathsf{T}_{G_2}[w] \leftarrow\$ \{0,1\}^{\lambda_2}$
return $\mathsf{T}_{G_2}[w]$
```

```
$O_{\mathsf{Sign}}(\mathsf{msg}, r)$ // $\mathcal{G}_{[0,2)}$
$i \leftarrow \mathsf{M}(\mathsf{msg}, r)$
$w \leftarrow O_H(\mathsf{msg}, r)$
$\alpha \leftarrow O_{G_1}(w) \oplus r$
$\gamma \leftarrow O_{G_2}(w)$
$\mathsf{sig} \leftarrow (0 \| w \| \alpha \| \gamma)^d \pmod N$
$\mathsf{T}_{\mathsf{Sign}}[i] \leftarrow \mathsf{sig}$
return $\mathsf{sig}$
```

```
$O_H(\mathsf{msg}, r)$ // $\mathcal{G}_{[0,2)}$
$i \leftarrow \mathsf{M}(\mathsf{msg}, r)$
if $\mathsf{T}_H[i] = \bot$ then
    $w \leftarrow\$ \{0,1\}^{\lambda_0}$
    $\mathsf{T}_H[i] \leftarrow w$
return $\mathsf{T}_H[i]$
```

```
$O_H(\mathsf{msg}, r)$ $O_{\mathsf{Sign}}(\mathsf{msg}, r)$ // $\mathcal{G}_{[2,4)}$
$i \leftarrow \mathsf{M}(\mathsf{msg}, r)$
if $\mathsf{T}_H[i] = \bot$ then
    $w \leftarrow\$ \{0,1\}^{\lambda_0}$
    $\mathsf{T}_H[i] \leftarrow w$
    if $\mathsf{T}_{G_1}[w] = \bot$ then
        $\alpha \leftarrow\$ \{0,1\}^{\lambda_1}$
        $\mathsf{T}_{G_1}[w] \leftarrow \alpha \oplus r$
    else $\alpha \leftarrow \mathsf{T}_{G_1}[w] \oplus r$
    if $\mathsf{T}_{G_2}[w] = \bot$ then
        $\gamma \leftarrow\$ \{0,1\}^{\lambda_2}$
        $\mathsf{T}_{G_2}[w] \leftarrow \gamma$
    else $\gamma \leftarrow \mathsf{T}_{G_2}[w]$
    $\mathsf{T}_\mathsf{y}[i] \leftarrow w \| \alpha \| \gamma$ // $\mathcal{G}_{[3,4)}$
else
    $w \leftarrow \mathsf{T}_H[i]$ // $\mathcal{G}_{[2,3)}$
    $\alpha \leftarrow \mathsf{T}_{G_1}[w] \oplus r$ // $\mathcal{G}_{[2,3)}$
    $\gamma \leftarrow \mathsf{T}_{G_2}[w]$ // $\mathcal{G}_{[2,3)}$
    $w \| \alpha \| \gamma \leftarrow \mathsf{T}_\mathsf{y}[i]$ // $\mathcal{G}_{[3,4)}$
    $\mathsf{T}_{G_1}[w] \leftarrow \alpha \oplus r$
    $\mathsf{T}_{G_2}[w] \leftarrow \gamma$
$\mathsf{sig} \leftarrow (0 \| w \| \alpha \| \gamma)^d \pmod N$
$\mathsf{T}_{\mathsf{Sign}}[i] \leftarrow \mathsf{sig}$
return $\mathsf{sig}$
return $w$
```

```
$O_H(\mathsf{msg}, r)$ $O_{\mathsf{Sign}}(\mathsf{msg}, r)$ // $\mathcal{G}_{[4,6)}$
$i \leftarrow \mathsf{M}(\mathsf{msg}, r)$
if $\mathsf{T}_H[i] = \bot$ then
    $w \leftarrow\$ \{0,1\}^{\lambda_0}$
    $\mathsf{T}_H[i] \leftarrow w$
    if $\mathsf{T}_{G_1}[w] \neq \bot$ then
        $\mathsf{bad} \leftarrow \mathsf{true}$
        $\alpha \leftarrow \mathsf{T}_{G_1}[w] \oplus r$ // $\mathcal{G}_{[4,5)}$
        $\alpha \leftarrow\$ \{0,1\}^{\lambda_1}$ // $\mathcal{G}_{[5,6)}$
    else $\alpha \leftarrow\$ \{0,1\}^{\lambda_1}$
    $\mathsf{T}_{G_1}[w] \leftarrow \alpha \oplus r$
    if $\mathsf{T}_{G_2}[w] \neq \bot$ then
        $\mathsf{bad} \leftarrow \mathsf{true}$
        $\gamma \leftarrow \mathsf{T}_{G_2}[w]$ // $\mathcal{G}_{[4,5)}$
        $\gamma \leftarrow\$ \{0,1\}^{\lambda_2}$ // $\mathcal{G}_{[5,6)}$
    else $\gamma \leftarrow\$ \{0,1\}^{\lambda_2}$
    $\mathsf{T}_{G_2}[w] \leftarrow \gamma$
    $\mathsf{T}_\mathsf{y}[i] \leftarrow w \| \alpha \| \gamma$
else
    $w \| \alpha \| \gamma \leftarrow \mathsf{T}_\mathsf{y}[i]$
    $\mathsf{T}_{G_1}[w] \leftarrow \alpha \oplus r$
    $\mathsf{T}_{G_2}[w] \leftarrow \gamma$
$\mathsf{sig} \leftarrow (0 \| w \| \alpha \| \gamma)^d \pmod N$
$\mathsf{T}_{\mathsf{Sign}}[i] \leftarrow \mathsf{sig}$
return $\mathsf{sig}$
return $w$
```

```
$O_H(\mathsf{msg}, r)$ $O_{\mathsf{Sign}}(\mathsf{msg}, r)$ // $\mathcal{G}_{[6,\infty)}$
$i \leftarrow \mathsf{M}(\mathsf{msg}, r)$
if $\mathsf{T}_H[i] = \bot$ then // $\mathcal{G}_{[6,7)}$
if $\mathsf{T}_\mathsf{y}[i] = \bot$ then // $\mathcal{G}_{[7,\infty)}$
    $\tilde{y} = w \| \alpha \| \gamma \leftarrow\$ \{0,1\}^{\lambda-1}$
    $\mathsf{T}_H[i] \leftarrow w$ // $\mathcal{G}_{[6,7)}$
    $\mathsf{T}_{G_1}[w] \leftarrow \alpha \oplus r$ // $\mathcal{G}_{[6,8)}$
    $\mathsf{T}_{G_2}[w] \leftarrow \gamma$ // $\mathcal{G}_{[6,8)}$
    $\mathsf{T}_\mathsf{y}[i] \leftarrow \tilde{y}$
else
    $\tilde{y} = w \| \alpha \| \gamma \leftarrow \mathsf{T}_\mathsf{y}[i]$
    $\mathsf{T}_{G_1}[w] \leftarrow \alpha \oplus r$ // $\mathcal{G}_{[6,8)}$
    $\mathsf{T}_{G_2}[w] \leftarrow \gamma$ // $\mathcal{G}_{[6,8)}$
$\mathsf{T}_{G_1}[w] \leftarrow \alpha \oplus r$ // $\mathcal{G}_{[8,\infty)}$
$\mathsf{T}_{G_2}[w] \leftarrow \gamma$ // $\mathcal{G}_{[8,\infty)}$
$\mathsf{sig} \leftarrow (0 \| \tilde{y})^d \pmod N$
$\mathsf{T}_{\mathsf{Sign}}[i] \leftarrow \mathsf{sig}$
return $\mathsf{sig}$
return $w$
```

Fig. 31. Games used in the proof of Lemma 1

```
$\mathcal{B}'^{O_{\mathsf{Chal}},O_{\mathsf{Corr}}}(\mathsf{pp}, N, e)$
  $\mathsf{ctr} \leftarrow 1$
  $(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}^{O_{\mathsf{Sign}},O_H,O_{G_1},O_{G_2}}(\mathsf{pp}, (N, e))$
  $(b^* \| w^* \| \alpha^* \| \gamma^*) \leftarrow (\mathsf{sig}^*)^e \pmod N$
  $r^* \leftarrow O_{G_1}(w^*) \oplus \alpha^*$
  $i^* \leftarrow \mathsf{M}[\mathsf{msg}^*, r^*]$
  $x' \leftarrow O_{\mathsf{Chal}}(i^*)$
  $x^* \leftarrow \mathsf{sig}^*$
  return $(x^*, i^*)$

$\mathsf{M}(\mathsf{msg}, r)$
  if $\mathsf{T}_{\mathsf{M}}[\mathsf{msg}, r] = \bot$ then
    $\mathsf{T}_{\mathsf{M}}[\mathsf{msg}, r] \leftarrow \mathsf{ctr}$
    $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
  return $\mathsf{T}_{\mathsf{M}}[\mathsf{msg}, r]$

$O_{\mathsf{Sign}}(\mathsf{msg}, r)$
  $i \leftarrow \mathsf{M}(\mathsf{msg}, r)$
  $0 \| w \| \alpha \| \gamma \leftarrow O_{\mathsf{Chal}}(i)$
  $\mathsf{T}_{G_1}[w] \leftarrow \alpha \oplus r$
  $\mathsf{T}_{G_2}[w] \leftarrow \gamma$
  $\mathsf{sig} \leftarrow O_{\mathsf{Corr}}(i)$
  return $\mathsf{sig}$

$O_H(\mathsf{msg}, r)$
  $i \leftarrow \mathsf{M}(\mathsf{msg}, r)$
  $0 \| w \| \alpha \| \gamma \leftarrow O_{\mathsf{Chal}}(i)$
  $\mathsf{T}_{G_1}[w] \leftarrow \alpha \oplus r$
  $\mathsf{T}_{G_2}[w] \leftarrow \gamma$
  return $w$

$O_{G_1}(w)$
  if $\mathsf{T}_{G_1}[w] = \bot$ then
    $\mathsf{T}_{G_1}[w] \leftarrow\$ \{0,1\}^{\lambda_1}$
  return $\mathsf{T}_{G_1}[w]$

$O_{G_2}(w)$
  if $\mathsf{T}_{G_2}[w] = \bot$ then
    $\mathsf{T}_{G_2}[w] \leftarrow\$ \{0,1\}^{\lambda_2}$
  return $\mathsf{T}_{G_2}[w]$
```

Fig. 32. Reduction used in the proof of Lemma 1

Transition from game $\mathcal{G}_1$ to $\mathcal{G}_2$. This transition makes multiple changes to both the O_H and O_{Sign} oracles. For O_H, when lazily sampling w for queries on new (msg, r), the oracle now also preemptively samples the $\mathsf{T}_{G_1}[w]$ and $\mathsf{T}_{G_2}[w]$ entries if not already assigned. Finally, the oracle reads α and γ from $\mathsf{T}_{G_1}[w]$ and $\mathsf{T}_{G_2}[w]$ and then writes them back into the tables when recovering w from T_H for a previously queried (msg, r). The game's behavior is unchanged by these modifications to O_H.

For O_{Sign}, we now explicitly implement the logic for updating $\mathsf{T}_{G_1}[w]$ and $\mathsf{T}_{G_2}[w]$ which were previously done by querying the O_{G_1} and O_{G_2} oracles. This code is only run when (msg, r) is not found in T_H (i.e., queried to neither O_H nor O_{Sign}); otherwise, existing values are read. This is valid because the change to O_H described above ensures α and γ are assigned whenever w is, so O_{Sign} only needs to perform this assignment for new (msg, r) queries. Overall, we have that

$$\Pr[\mathcal{G}_1(\lambda) \Rightarrow 1] = \Pr[\mathcal{G}_2(\lambda) \Rightarrow 1].$$

Transition from game $\mathcal{G}_2$ to $\mathcal{G}_3$. This transition introduces a table T_{y} which stores $w \| \alpha \| \gamma$ at index $i \leftarrow \mathsf{M}[\mathsf{msg}, r]$. Both O_H and O_{Sign} now write to this table (in addition to T_H, T_{G_1}, and T_{G_2}). If (msg, r) is found in T_H (i.e., queried to either O_H or O_{Sign}), the oracles read the combined $w \| \alpha \| \gamma$ from T_{y} rather than reading these values from separate tables. This game and the previous are identical, so

$$\Pr[\mathcal{G}_2(\lambda) \Rightarrow 1] = \Pr[\mathcal{G}_3(\lambda) \Rightarrow 1].$$

Transition from game $\mathcal{G}_3$ to $\mathcal{G}_4$. This transition performs a straightforward rewrite for the lazy sampling of G_1 and G_2 for new (msg, r) queries in both O_H and O_{Sign}. A bad flag is also added, set to true if the newly sampled w for new (msg, r) corresponds to an already defined entry in T_{G_1} or T_{G_2}. Because these changes are merely a rewriting, the games are equivalent, hence

$$\Pr[\mathcal{G}_3(\lambda) \Rightarrow 1] = \Pr[\mathcal{G}_4(\lambda) \Rightarrow 1].$$

Transition from game $\mathcal{G}_4$ to $\mathcal{G}_5$. This transition changes the code following the bad flags. For a sampled w from a new (msg, r) query, the oracle now samples new α and γ and overwrites $\mathsf{T}_{G_1}[w]$ and $\mathsf{T}_{G_2}[w]$, even if they are already defined. This bad event (i.e., the entries being predefined) occurs if a

w sampled during any of the $q_S + q_H$ queries to O_{Sign} and O_H or during the O_H in verification was previously sampled and stored into T_H. This occurs if w was

1. sampled during a previous adversarial query to O_{Sign} or O_H,
2. previously queried to O_{G_1} or O_{G_2} as one of the q_G queries,
3. sampled during the O_{G_1} run in verification to extract r^*.

Because w is sampled uniformly and λ_0 bits, and the games are identical-until-bad, we have that

$$\Pr[\mathcal{G}_4(\lambda) \Rightarrow 1] - \Pr[\mathcal{G}_5(\lambda) \Rightarrow 1] \leq (q_S + q_H + 1)(q_S + q_H + q_G + 1) \cdot 2^{-\lambda_0}.$$

Transition from game $\mathcal{G}_5$ to $\mathcal{G}_6$. This transition simplifies the code. Per the previous change, w, α, and γ are now sampled simultaneously, so we rewrite them as a single variable $\tilde{y} = w\|\alpha\|\gamma$. This $\tilde{y}$ is sampled, stored in T_{y}, and used in the signature. Note that we still store $\alpha \oplus r$ and γ in T_{G_1} and T_{G_2} respectively to handle queries to the O_{G_1} and O_{G_2} oracles. This game and the previous are identical, so

$$\Pr[\mathcal{G}_5(\lambda) \Rightarrow 1] = \Pr[\mathcal{G}_6(\lambda) \Rightarrow 1].$$

Transition from game $\mathcal{G}_6$ to $\mathcal{G}_7$. This transition removes T_H, as it is redundant. In particular, T_H is only written to and used to check if $\mathsf{T}_H[i] = \perp$ for $i \leftarrow \mathsf{M}(\mathsf{msg}, r)$. Because T_{y} is assigned at index i exactly when T_H is, this game performs this check on T_{y} instead and drops T_H entirely. Because these changes are merely a rewriting, the games are equivalent, hence

$$\Pr[\mathcal{G}_6(\lambda) \Rightarrow 1] = \Pr[\mathcal{G}_7(\lambda) \Rightarrow 1].$$

Transition from game $\mathcal{G}_7$ to $\mathcal{G}_8$. This transition observes that T_{G_1} and T_{G_2} are assigned in both branches of the conditionals in O_{Sign} and O_H, so the assignment code is moved outside the conditional. This is a simple rewriting, so the game is unchanged.

$$\Pr[\mathcal{G}_7(\lambda) \Rightarrow 1] = \Pr[\mathcal{G}_8(\lambda) \Rightarrow 1].$$

Game $\mathcal{G}_8$. This game is equivalent to reduction $\mathcal{B}'$ shown in Figure 32 playing the modified RSA game $\mathcal{G}^{\mathsf{mc\text{-}RSA^*\text{-}ac}}$ game. The T_{y} and $\mathsf{T}_{\mathsf{Sign}}$ tables in $\mathcal{G}_7$ are equivalent to the T_{y} and T_{x} tables in $\mathcal{G}^{\mathsf{mc\text{-}RSA^*\text{-}ac}}$, respectively. Furthermore, the sampling of $\tilde{y}$ corresponds to the O_{Chal} oracle, and $(0\|\tilde{y})^d \pmod N$ corresponds to the O_{Corr} oracle. The final O_H query in $\mathcal{G}_8$ ensures that the final O_{G_1} and O_{G_2} queries are consistent with the challenge in the final O_{Chal} query of $\mathcal{B}'$. It follows that

$$\Pr[\mathcal{G}_7(\lambda) \Rightarrow 1] = \mathbf{Adv}^{\mathsf{mc\text{-}RSA^*\text{-}ac}}_{\mathcal{B}'}(\lambda).$$

We can now apply Lemma 2 to bound this advantage. Because $\mathcal{B}'$ queries O_{Chal} once during each O_{Sign} and O_H query and makes one additional on the adversary's $\mathcal{A}$ forgery then $\mathcal{B}'$ makes $q_S + q_H + 1$ challenge queries. Furthermore, the ctr maintained and assigned during M cannot be incremented more than $q_S + q_H + 1$ because M is also run once during each O_{Sign} and O_H query or when processing the forgery, thus the reduction $\mathcal{B}'$ is admissible. We have that there exists a PPT algorithm $\mathcal{B}$ such that

$$\mathbf{Adv}^{\mathsf{mc\text{-}RSA^*\text{-}ac}}_{\mathcal{B}'}(\lambda) \leq 2(q_S + q_H + 1) \cdot \mathbf{Adv}^{\mathsf{RSA}}_{\mathcal{B}}(\lambda) + (q_S + q_H + 1)^2 \cdot 2^{-\lambda - 1}.$$

The theorem bound follows from these observations. □

```
$\mathcal{G}_n(\lambda)$ for $n \in \{0,1\}$
    $S \leftarrow \varnothing$
    $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
    $\mathsf{st} \leftarrow \mathsf{Prim.Init}(\mathsf{pp})$
    $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
    $(\mathsf{k}, \mathsf{st}_{\mathsf{Enc}}, \mathsf{st}_{\mathsf{Dec}}) \leftarrow \mathsf{prE.KeyGen}(\mathsf{pp})$
    $\mathsf{dk} \leftarrow (\mathsf{vk}, \mathsf{k})$
    $(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}^{O^{\mathcal{A}}_{\mathsf{Sign}}, O^{\mathcal{A}}_{\mathsf{aSign}}, O^{\mathcal{A}}_{\mathsf{LS}}, O^{\mathcal{A}}_{\mathsf{Prog}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{dk}, \mathsf{st}_{\mathsf{Dec}})$
    $b_{\mathsf{valid}} \leftarrow [\![\mathsf{S.Vrfy}(\mathsf{vk}, \mathsf{msg}^*, \mathsf{sig}^*) = 1]\!]$
    $b_{\mathsf{new}} \leftarrow [\![(\mathsf{msg}^*, \mathsf{sig}^*) \notin S]\!]$
    $\mathbf{return}\ b_{\mathsf{valid}} \wedge b_{\mathsf{new}}$

$O^{\mathcal{A}}_{\mathsf{LS}}(x)$
    $\mathbf{return}\ \mathsf{Prim.LS}(x : \mathsf{st})$
```

```
$O^{\mathcal{A}}_{\mathsf{Sign}}(\mathsf{msg})$
    $\mathsf{sig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg})$
    $S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{sig})\}$
    $\mathbf{return}\ \mathsf{sig}$

$O^{\mathcal{A}}_{\mathsf{aSign}}(\mathsf{msg}, \mathsf{amsg})$
    $\mathsf{act} \leftarrow \mathsf{prE.Enc}^{O^{\mathcal{A}}_{\mathsf{LS}}}(\mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{st}_{\mathsf{Enc}})$    // $\mathcal{G}_{[0,1)}$
    $\mathsf{act} \leftarrow\!\!\$\ \mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}$    // $\mathcal{G}_{[1,\infty)}$
    $\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}; \mathsf{act})$
    $\mathsf{SE.Sim}^{O^{\mathcal{A}}_{\mathsf{LS}}, O^{\mathcal{A}}_{\mathsf{Prog}}}(\mathsf{pp}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg}, \mathsf{act} : \mathsf{st})$    // $\mathcal{G}_{[1,\infty)}$
    $S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{asig})\}$
    $\mathbf{return}\ \mathsf{asig}$

$O^{\mathcal{A}}_{\mathsf{Prog}}(z)$
    $\mathbf{return}\ \mathsf{Prim.Prog}(z : \mathsf{st})$
```

Fig. 33. Games used in the proof of Theorem 8

E.3 Full Proof of Theorem 8 (RRep* with SIM-\$CT Encryption is RUF-CAMA)

In Section 5, we argued that RRep*, when instantiated with standard chosen message (SUF-CMA)-secure signature schemes and pseudorandom encryption schemes that are SIM-\$CT secure, produce recipient unforgeable anamorphic signature schemes. We provide a full proof here.

Theorem 8 (RRep* with SIM-\$CT Encryption is RUF-CAMA). *Let* S *be an unforgeable publicly randomness-recovering signature scheme and* prE *be a pseudorandom encryption scheme constructed from ideal primitive* Prim *that is simulatable with random ciphertexts. Then* $\mathsf{aS} = \mathsf{RRep}^\star[\mathsf{S}, \mathsf{prE}]$ *(Construction 1) is recipient unforgeable. In particular, for all PPT adversaries $\mathcal{A}$, there exist PPT adversaries $\mathcal{B}_0$ and $\mathcal{B}_1$ such that*

$$\mathbf{Adv}^{\mathsf{RUF\text{-}CAMA}(O_{\mathsf{LS}}, O_{\mathsf{Prog}})}_{\mathsf{aS},\mathsf{Prim},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{SIM\text{-}\$CT}}_{\mathsf{prE},\mathsf{Prim},\mathcal{B}_0}(\lambda) + \mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{S},\mathcal{B}_1}(\lambda).$$

Note that our proof will require that S makes no queries to the ideal primitive.

Proof. The proof follows from a single game hop from $\mathcal{G}_0$, which is equivalent to $\mathcal{G}^{\mathsf{RUF\text{-}CAMA}(O_{\mathsf{LS}}, O_{\mathsf{Prog}})}$, to a hybrid $\mathcal{G}_1$. A PPT adversary against $\mathcal{G}_1$ can then be used to construct a PPT algorithm against the SUF-CMA security of S. Games $\mathcal{G}_0$ through $\mathcal{G}_2$ are shown in Figure 33.

Game $\mathcal{G}_0$. This game is equivalent to the RUF-CAMA security game over $\mathsf{aS} = \mathsf{RRep}^\star[\mathsf{S}, \mathsf{prE}]$ and ideal primitive Prim underlying prE, where the adversary additionally has access to O_{LS} and O_{Prog} for Prim. To arrive at $\mathcal{G}_0$, we substitute in the definition of the randomness replacement transform RRep*.

Transition from game $\mathcal{G}_0$ to $\mathcal{G}_1$. This transition modifies the anamorphic signing oracle $O^{\mathcal{A}}_{\mathsf{aSign}}$ by swapping act from a pseudorandom encryption of amsg to a random sample from $\mathsf{prE}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}$. Game $\mathcal{G}_1$ then calls SE.Sim to attempt to make act look consistent with amsg. Distinguishing between $\mathcal{G}_0$ and $\mathcal{G}_1$ is bounded by the simulatability with random ciphertext advantage.

Concretely, let $\mathcal{A}$ be a PPT algorithm that plays either game $\mathcal{G}_0$ or game $\mathcal{G}_1$. Then we can construct a PPT algorithm $\mathcal{B}_0$ against the SIM-\$CT security of prE. We first review the games that $\mathcal{A}$ and $\mathcal{B}_0$ play. The adversary $\mathcal{A}$ receives pp, vk, dk, and $\mathsf{st}_{\mathsf{Dec}}$ as well as access to a signing oracle $O^{\mathcal{A}}_{\mathsf{Sign}}$, anamorphic signing oracle $O^{\mathcal{A}}_{\mathsf{aSign}}$, and primitive oracles $O^{\mathcal{A}}_{\mathsf{LS}}$ and $O^{\mathcal{A}}_{\mathsf{Prog}}$ and attempts to output a forgery $(\mathsf{msg}^*, \mathsf{sig}^*)$.

The reduction $\mathcal{B}_0$ plays the $\mathcal{G}^{\mathsf{SIM\text{-}\$CT}}$ game on prE where it receives pp, k, and $\mathsf{st}_{\mathsf{Dec}}$ and has access to an encryption oracle $O^{\mathcal{B}_0}_{\mathsf{Enc}}$ which outputs either honest encryptions or randomly sampled ciphertexts depending on a random bit b as well as primitive oracles $O^{\mathcal{A}}_{\mathsf{LS}}$ and $O^{\mathcal{A}}_{\mathsf{Prog}}$.

Upon initialization, $\mathcal{B}_0$ receives pp, k, and $\mathsf{st}_{\mathsf{Dec}}$ and internally generates a signing keypair $(\mathsf{vk}, \mathsf{sk})$ using pp. It then runs $\mathcal{A}$ with pp, vk, and $\mathsf{dk} = \mathsf{k}$ as input. To respond to $\mathcal{A}$'s signing queries to $O^{\mathcal{A}}_{\mathsf{Sign}}$, the reduction $\mathcal{B}_0$ generates a signature using sk which it sends to $\mathcal{A}$. When $\mathcal{A}$ makes an anamorphic signing query to $O^{\mathcal{A}}_{\mathsf{aSign}}$ on $(\mathsf{msg}, \mathsf{amsg})$, the reduction $\mathcal{B}_0$ queries its encryption oracle $O^{\mathcal{B}}_{\mathsf{Enc}}$ on $(\mathsf{msg}, \mathsf{amsg})$, receives a ciphertext act, and computes $\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}; \mathsf{act})$. It sends asig to $\mathcal{A}$. Finally, $\mathcal{B}_0$ simply forwards queries to $O^{\mathcal{A}}_{\mathsf{LS}}$ and $O^{\mathcal{A}}_{\mathsf{Prog}}$ to its own $O^{\mathcal{B}}_{\mathsf{LS}}$ and $O^{\mathcal{B}}_{\mathsf{Prog}}$ respectively. When $\mathcal{A}$ terminates and outputs a forgery $(\mathsf{msg}^*, \mathsf{sig}^*)$, then $\mathcal{B}$ outputs 1 if the forgery is valid and new. Observe that when $b = 0$, $\mathcal{B}_0$ simulates $\mathcal{G}_0$ to $\mathcal{A}$, while when $b = 1$, $\mathcal{B}_0$ simulates $\mathcal{G}_1$ to $\mathcal{A}$. This required that $\mathsf{S.Sign}$ not query Prim so that swapping the order of $\mathsf{S.Sign}$ and $\mathsf{SE.Sim}$ has no effect. Furthermore, $\mathcal{B}_0$ is PPT because $\mathcal{A}$ is PPT, $\mathsf{S.KeyGen}$ is PPT, and $\mathsf{S.Sign}$ is PPT. It follows that

$$\Pr[\mathcal{G}_0(\lambda) \Rightarrow 1] - \Pr[\mathcal{G}_1(\lambda) \Rightarrow 1] = \mathbf{Adv}^{\mathsf{SIM\text{-}\$CT}}_{\mathsf{prE},\mathsf{Prim},\mathcal{B}_0}(\lambda).$$

Game $\mathcal{G}_1$. Winning this game is bounded by the unforgeability of S. Concretely, let $\mathcal{A}$ be a PPT algorithm against $\mathcal{G}_1$. Then we can construct a PPT algorithm $\mathcal{B}_1$ against the $\mathsf{SUF\text{-}CMA}$ security of S. We first review the games that $\mathcal{A}$ and $\mathcal{B}_1$ play. The adversary $\mathcal{A}$ plays game $\mathcal{G}_1$ and receives pp, vk, dk, and $\mathsf{st}_{\mathsf{Dec}}$ as well as access to a signing oracle $O^{\mathcal{A}}_{\mathsf{Sign}}$, anamorphic signing oracle $O^{\mathcal{A}}_{\mathsf{aSign}}$, and primitive oracles $O^{\mathcal{A}}_{\mathsf{LS}}$ and $O^{\mathcal{A}}_{\mathsf{Prog}}$. The reduction $\mathcal{B}_1$ plays the $\mathcal{G}^{\mathsf{SUF\text{-}CMA}}$ game on S where it receives pp, vk, and access to a signing oracle $O^{\mathcal{B}_1}_{\mathsf{Sign}}$.

We construct $\mathcal{B}_1$ as follows. Upon initialization, $\mathcal{B}_1$ generates $\mathsf{dk}, \mathsf{st}_{\mathsf{Enc}}, \mathsf{st}_{\mathsf{Dec}}$ for prE and st for Prim. When answering $\mathcal{A}$'s signing queries to $O^{\mathcal{A}}_{\mathsf{Sign}}$, $\mathcal{B}_1$ simply forwards the query to its own signing oracle $O^{\mathcal{B}_1}_{\mathsf{Sign}}$. To respond to $\mathcal{A}$'s primitive oracle queries, $\mathcal{B}$ simply runs $\mathsf{Prim.LS}$ or $\mathsf{Prim.Prog}$. When $\mathcal{A}$ queries $(\mathsf{msg}, \mathsf{amsg})$ to its anamorphic signing oracle $O^{\mathcal{A}}_{\mathsf{aSign}}$, the reduction $\mathcal{B}_1$ queries its own signing oracle $O^{\mathcal{B}_1}_{\mathsf{Sign}}$ on msg to receive sig, calls $\mathsf{S.RRecov}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$ (which is possible since S is publicly randomness recovering) to obtain randomness act, and calls $\mathsf{SE.Sim}(\mathsf{pp}, \mathsf{k}, \mathsf{msg}, \mathsf{amsg}, \mathsf{act} : \mathsf{st}_{\mathsf{Enc}})$. Observe that act is sampled uniformly from $\mathsf{prE}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}$ in $\mathcal{G}_1$, which is equivalent to running the honest signing process.

It follows that $\mathcal{B}_1$ simulates $\mathcal{G}_1$ to $\mathcal{A}$. Furthermore, any $(\mathsf{msg}, \mathsf{sig})$ pair $\mathcal{A}$ has received has also been received by $\mathcal{B}_1$, thus any forgery $\mathcal{A}$ outputs is a forgery for S. Finally, $\mathcal{B}_1$ is PPT because $\mathcal{A}$ is PPT, Prim operations are PPT, $\mathsf{S.RRecov}$ is PPT, and $\mathsf{SE.Sim}$ is PPT. It follows that

$$\Pr[\mathcal{G}_1(\lambda) \Rightarrow 1] = \mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{S},\mathcal{B}_1}(\lambda).$$

The theorem bound follows from these observations. □

E.4 Full Proof of Corollary 5 (RRep*[BB, RHtX] is RUF-CAMA)

In Section 5.6, we argued that the randomized hash then XOR encryption scheme RHtX was simulatable with random ciphertexts. Here, we review the high level, specify it in our general ideal primitive notation, and prove its simulatability.

Recall that the scheme operates on λ-bit keys and messages and outputs 2λ-bit ciphertexts. Encryption $\mathsf{RHtX.Enc}(\mathsf{k}, \mathsf{msg})$ outputs $\mathsf{ct} \leftarrow r \| (\mathsf{msg} \oplus H(\mathsf{k} \| r))$ where r is a fresh random λ-bit string and $H : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$ is a hash function. Decryption works by extracting r, computing $H(\mathsf{k} \| r)$, and retrieving the message msg. Our analysis will treat the primitive H as a random oracle. We will instantiate H using lazy sampling i.e. H is defined by a table mapping inputs to outputs. When queried on new inputs, the output is freshly sampled, otherwise the output stored in the table is returned. When H is programmed on an input previously queried we elect to explicitly reject the programming. Figure 34 formally specifies H as well as the simulator $\mathsf{RHtX.Sim}$. We now prove that RHtX is simulatable with random ciphertexts.

```
$H.\mathsf{LS}(x : \mathsf{st})$
    if $\mathsf{st}[x] = \perp$
        $\mathsf{st}[x] \leftarrow\!\!\$ \{0,1\}^\lambda$
    return $\mathsf{st}[x]$

$H.\mathsf{Prog}(z : \mathsf{st})$
    $(x, y) \leftarrow z$
    if $\mathsf{st}[x] \neq \perp$
        return $\perp$
    $\mathsf{st}[x] \leftarrow y$
    return $\top$

$\mathsf{RHtX.Enc}^{O_{\mathsf{LS}}}(\mathsf{pp}, \mathsf{k}, \mathsf{msg})$
    $r \leftarrow\!\!\$ \{0,1\}^\lambda$
    $x \leftarrow \mathsf{k} \| r$
    $y \leftarrow O_{\mathsf{LS}}(x)$
    $\mathsf{ct} \leftarrow r \| (y \oplus \mathsf{msg})$
    return $\mathsf{ct}$

$\mathsf{RHtX.Sim}^{O_{\mathsf{LS}}, O_{\mathsf{Prog}}}(\mathsf{pp}, \mathsf{k}, \mathsf{msg}, \mathsf{ct} : \mathsf{st}_{\mathsf{Enc}})$
    $r \| \gamma \leftarrow \mathsf{ct}$
    $x \leftarrow \mathsf{k} \| r$
    $y \leftarrow \gamma \oplus \mathsf{msg}$
    $O_{\mathsf{Prog}}((x, y))$
    return
```

```
$\mathcal{G}_n(\lambda)$ for $n \in \{0,1,2\}$
    $b \leftarrow\!\!\$ \{0,1\}$
    $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
    $\mathsf{st} \leftarrow \mathsf{Prim.Init}(\mathsf{pp})$
    $\mathsf{k} \leftarrow \mathsf{SE.KeyGen}(\mathsf{pp})$
    $b^* \leftarrow \mathcal{A}^{O^{\mathcal{A}}_{\mathsf{Enc}}, O^{\mathcal{A}}_{\mathsf{LS}}, O^{\mathcal{A}}_{\mathsf{Prog}}}(\mathsf{pp}, \mathsf{k})$
    return $[\![b = b^*]\!]$

$O^{\mathcal{A}}_{\mathsf{LS}}(x)$
    if $\mathsf{st}[x] = \perp$
        $\mathsf{st}[x] \leftarrow\!\!\$ \{0,1\}^\lambda$
    return $\mathsf{st}[x]$

$O^{\mathcal{A}}_{\mathsf{Prog}}(z)$
    $(x, y) \leftarrow z$
    if $\mathsf{st}[x] \neq \perp$
        return $\perp$
    $\mathsf{st}[x] \leftarrow y$
    return $\top$
```

```
$O^{\mathcal{A}}_{\mathsf{Enc}}(\mathsf{msg})$
    if $b = 0$ then
        $r \leftarrow\!\!\$ \{0,1\}^\lambda$
        $x \leftarrow \mathsf{k} \| r$
        $y \leftarrow O_{\mathsf{LS}}(x)$
        $\mathsf{ct} \leftarrow r \| (y \oplus \mathsf{msg})$
    else
        $\mathsf{ct} \leftarrow\!\!\$ \{0,1\}^{2\lambda}$    // $\mathcal{G}_{[0,1)}$
        $r \| \gamma \leftarrow \mathsf{ct}$    // $\mathcal{G}_{[0,1)}$
        $r \leftarrow\!\!\$ \{0,1\}^\lambda$    // $\mathcal{G}_{[1,\infty)}$
        $x \leftarrow \mathsf{k} \| r$
        $y \leftarrow \gamma \oplus \mathsf{msg}$    // $\mathcal{G}_{[0,1)}$
        $y \leftarrow\!\!\$ \{0,1\}^\lambda$    // $\mathcal{G}_{[1,2)}$
        $O_{\mathsf{Prog}}((x, y))$    // $\mathcal{G}_{[0,2)}$
        $y \leftarrow O_{\mathsf{LS}}(x)$    // $\mathcal{G}_{[2,\infty)}$
        $\mathsf{ct} \leftarrow r \| (y \oplus \mathsf{msg})$    // $\mathcal{G}_{[1,\infty)}$
    return $\mathsf{ct}$
```

Fig. 34. Lazy sampling, programming, and simulation for H and RHtX (left) and games used in the proof of Lemma 3 (right)

Lemma 3 (RHtX is SIM-\$CT). *Randomized hash then XOR is simulatable with random ciphertexts, provided the hash function H underlying it is modeled as a random oracle. In particular, for all PPT adversaries $\mathcal{A}$ that make q_P programming queries, q_H random oracle queries, and q_E encryption queries,*

$$\mathbf{Adv}^{\mathsf{SIM\text{-}\$CT}}_{\mathsf{RHtX},H,\mathcal{A}}(\lambda) \leq \frac{q_E(q_P + q_H + q_E)}{2^\lambda}.$$

Proof. The proof follows from a series of game hops from $\mathcal{G}_0$, equivalent to the $\mathcal{G}^{\mathsf{SIM\text{-}\$CT}}$ game on RHtX, to $\mathcal{G}_2$, which no PPT adversary can win with probability greater than $\frac{1}{2}$. The games are shown in Figure 34.

Game $\mathcal{G}_0$. This game is equivalent to the $\mathcal{G}^{\mathsf{SIM\text{-}\$CT}}$ game over RHtX built on random oracle H. That is, $\mathcal{G}_0 = \mathcal{G}^{\mathsf{SIM\text{-}\$CT}}_{\mathsf{RHtX},H,\mathcal{A}}$. To arrive at $\mathcal{G}_0$ as it is notated, we substitute in the definition of RHtX and H as given in Figure 34 into $O^{\mathcal{A}}_{\mathsf{Enc}}$, $O^{\mathcal{A}}_{\mathsf{LS}}$, and $O^{\mathcal{A}}_{\mathsf{Prog}}$.

Transition from game $\mathcal{G}_0$ to $\mathcal{G}_1$. This transition samples r and y freshly, and returns $\mathsf{ct} = r \| (y \oplus \mathsf{msg})$ rather than directly sampling ct freshly. Games $\mathcal{G}_0$ and $\mathcal{G}_1$ are identical as r has remained randomly sampled and instead of setting y to msg masked with a random γ, we now set γ (implicitly returned in ct) to msg masked with a random y. Thus,

$$\Pr[\mathcal{G}_1(\lambda) \Rightarrow 1] = \Pr[\mathcal{G}_0(\lambda) \Rightarrow 1].$$

```
$\mathcal{B}^{O^{\mathcal{B}}_{\mathsf{Sign}}}(\mathsf{pp}, \mathsf{vk})$
    $\mathsf{dk} \leftarrow \mathsf{prF.KeyGen}(\mathsf{pp})$
    $\mathsf{st}_{\mathsf{aEnc}} \leftarrow 1$
    $(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}^{O^{\mathcal{A}}_{\mathsf{Sign}}, O^{\mathcal{A}}_{\mathsf{aSign}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{dk}, 1)$
    $\mathbf{return}\ (\mathsf{msg}^*, \mathsf{sig}^*)$
```

```
$O^{\mathcal{A}}_{\mathsf{Sign}}(\mathsf{msg})$
    $r \leftarrow\$ \mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}$
    $\mathsf{sig} \leftarrow O^{\mathcal{B}}_{\mathsf{Sign}}(\mathsf{msg}, r)$
    $\mathbf{return}\ \mathsf{sig}$
```

```
$O^{\mathcal{A}}_{\mathsf{aSign}}(\mathsf{msg}, \mathsf{amsg})$
    $r \leftarrow \mathsf{prF}(\mathsf{dk}, (\mathsf{ctr}_{\mathsf{aEnc}}, \mathsf{msg}, \mathsf{amsg}))$
    $\mathsf{asig} \leftarrow O^{\mathcal{B}}_{\mathsf{Sign}}(\mathsf{msg}, r)$
    $\mathsf{ctr}_{\mathsf{aEnc}} \leftarrow \mathsf{ctr}_{\mathsf{aEnc}} + 1$
    $\mathbf{return}\ \mathsf{asig}$
```

Fig. 35. Reduction used in the proof of Theorem 9

Transition from game $\mathcal{G}_1$ to $\mathcal{G}_2$. This transition replaces the random sampling and programming of x to output y to calling the lazy sampling oracle $O^{\mathcal{A}}_{\mathsf{LS}}(x)$. The two games $\mathcal{G}_1$ and $\mathcal{G}_2$ are identical except in the case where $\mathsf{st}[x]$ has already been assigned, which we consider as our bad event. Since r is randomly sampled, bad only occurs if r collides with a prior of $\mathcal{A}$'s q_P programming oracle queries, q_H hash queries, or q_E encryption queries and $b = 1$. By the fundamental lemma of game playing and a union bound,

$$|\Pr[\mathcal{G}_2(\lambda) \Rightarrow 1] - \Pr[\mathcal{G}_1(\lambda) \Rightarrow 1]| = \Pr[\mathsf{bad}] \leq \frac{q_E(q_P + q_H + q_E)}{2^{\lambda+1}}.$$

Game $\mathcal{G}_2$. Observe that $\mathcal{G}_2$ (where $b = 1$ or when $b = 0$) is simply running honest encryptions of RHtX. Thus $\mathcal{G}_2$ is identical when $b = 0$ or $b = 1$, so any adversary has at most a $\frac{1}{2}$ probability of winning.

The theorem bound follows from these observations. □

E.5 Full Proof of Theorem 9 (RIdP* with SUF-CRA Signatures is RUF-CAMA)

In Section 5, we argued that $\mathsf{RIdP}^\star$, when instantiated with chosen randomness unforgeable signature schemes, produce recipient unforgeable anamorphic signature schemes. We provide a full proof here.

Theorem 9 (RIdP* with SUF-CRA Signatures is RUF-CAMA). *Let S be a chosen-randomness unforgeable signature scheme and prF be a pseudorandom function. Then $\mathsf{aS} = \mathsf{RIdP}^\star[\mathsf{S}, \mathsf{prF}]$ (Construction 2) is recipient unforgeable. In particular, for all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{S},\mathcal{B}}(\lambda).$$

Proof. Let $\mathcal{A}$ be a PPT recipient forger for aS. We will use $\mathcal{A}$ to construct a PPT adversary $\mathcal{B}$ that breaks the $\mathsf{SUF\text{-}CRA}$-security of S.

We construct $\mathcal{B}$ from $\mathcal{A}$ as shown in Figure 35. Upon initialization, $\mathcal{B}$ generates $\mathsf{dk} \leftarrow \mathsf{prF.KeyGen}(\mathsf{pp})$ and gives vk, dk, and $\mathsf{st}_{\mathsf{aDec}} = 1$ to $\mathcal{A}$. When $\mathcal{A}$ queries $O^{\mathcal{A}}_{\mathsf{Sign}}$ with msg, $\mathcal{B}$ samples some randomness r and queries its signing oracle $O^{\mathcal{B}}_{\mathsf{Sign}}$ on (msg, r). When $\mathcal{A}$ queries $O^{\mathcal{A}}_{\mathsf{aSign}}$ with $(\mathsf{msg}, \mathsf{amsg})$ for an anamorphic signature, $\mathcal{B}$ computes $r \leftarrow \mathsf{prF}(\mathsf{dk}, (\mathsf{ctr}_{\mathsf{aEnc}}, \mathsf{msg}, \mathsf{amsg}))$ and queries (msg, r) to its signing oracle $O^{\mathcal{B}}_{\mathsf{Sign}}$, receives a signature sig, and forwards sig to $\mathcal{A}$. Once $\mathcal{A}$ outputs a forgery $(\mathsf{msg}^*, \mathsf{sig}^*)$, $\mathcal{B}$ forwards this as its forgery. From $\mathcal{A}$'s perspective, it has exactly played the recipient unforgeability game $\mathcal{G}^{\mathsf{RUF\text{-}CAMA}}$ on aS, hence $\mathcal{B}$'s advantage is identical to $\mathcal{A}$'s advantage. Finally, $\mathcal{B}$ is PPT because $\mathcal{A}$ is PPT, $\mathsf{prF.KeyGen}$ is PPT, and evaluating prF is PPT. The theorem bound follows from these observations. □