Anamorphic NIZK: Definition, Construction and More

Wenhui Kong[1], Gongxian Zeng[2], and Zhengan Huang[3]

[1] Tsinghua University, China
[2] Sun Yat-sen University, China
[3] Guangzhou University, China
kongwh22_thu@163.com, gxzeng@cs.hku.hk, zhahuang.sjtu@gmail.com

Abstract. Anamorphic encryption enables covert communication despite disclosure of ordinary decryption keys [16]. Work on anamorphic signatures extends this capability to settings permitting only authentication and shows that Fiat-Shamir preserves anamorphism [13,10]. However, this result assumes an already anamorphic protocol and does not by itself ensure witness privacy against the recipient, motivating anamorphism directly in non-interactive zero-knowledge proofs (NIZKs).
We introduce anamorphic NIZK (ANIZK), in which publicly verifiable proofs carry covert messages recoverable with a double key. The framework formalizes anamorphic security, strong robustness, and recipient anamorphic zero knowledge and simulation extractability. A generic construction in the random oracle model combines suitable Sigma protocols with authenticated encryption with associated data through a salted Fiat-Shamir transform. Applying ANIZK within the Naor-Yung paradigm [15] yields recipient anamorphic encryption satisfying STE-xCCA1, SIND-aCCA1, and SIND-nCCA security under the notions of [7]. Finally, an asymmetric variant of ANIZK, termed ASY-ANIZK, combines a labeled NIZK with public-key encryption satisfying ciphertext pseudorandomness and random ciphertext rejection. It separates public embedding from secret recovery and achieves weak robustness, as public embedding precludes strong robustness.

1 Introduction

Cryptography is typically designed to protect communication against an adversary who can observe or interfere with the resulting cryptographic objects. Anamorphic cryptography considers a more stringent setting in which the adversary, or *dictator*, also controls the cryptographic infrastructure and may obtain the users' secret keys. Persiano, Phan, and Yung [16] introduced *anamorphic encryption* to address this setting. Two parties sharing a pre-established secret called a *double key* can embed a covert message into an otherwise ordinary ciphertext. Under the ordinary secret key, the ciphertext decrypts to the prescribed message as usual, while the holder of the double key can additionally recover the covert message. At the same time, the ciphertext must remain computationally

indistinguishable from an honestly generated ciphertext to the dictator, who may possess the ordinary decryption key. Thus, the security goal is stronger than hiding the covert message: the existence of the covert channel itself must remain undetectable.

Anamorphism can persist even when encryption is unavailable. Kutyłowski et al. [13] considered a setting in which communication is essentially in the clear and signatures are the only cryptographic objects permitted by the dictator. Since the ordinary message is fully visible, the covert channel must instead be embedded in the signature. To achieve this, they introduced *anamorphic signature*, a primitive that allows a signer holding a double key to generate an ordinary-looking signature that authenticates the prescribed public message while simultaneously carrying a covert message recoverable by the some recipient.

The construction of anamorphic signatures points to a deeper connection with proof systems. In particular, Kutyłowski et al. [13] showed that the Fiat-Shamir transform [10] preserves anamorphism: whenever the underlying three-message public-coin protocol admits an anamorphic mode, the resulting Fiat-Shamir signature scheme does as well. This result suggests that anamorphism at the signature level can arise from a structural property of the underlying proof-generation procedure, rather than from an ad-hoc modification to the signing process.

However, this Fiat-Shamir result is inherently conditional. It assumes that the underlying interactive protocol is already anamorphic and only shows that anamorphism is preserved in the resulting signature. It does not explain how the anamorphic capability can be realized at the proof level. In fact, the construction of Kutyłowski et al. [13] generates the double key from the witness, and in their Fiat-Shamir construction the double key contains the signing key, and hence the witness. This is particularly problematic in the NIZK setting, where the witness is itself a protected secret.

This observation naturally raises the question of whether anamorphism can be realized directly at the level of non-interactive zero-knowledge proofs (NIZKs). The connection is particularly natural because NIZK simulation already appears as a structural ingredient in existing anamorphic constructions. For example, the anamorphic realization of the Naor-Yung paradigm [15] relies on the simulator of a simulation-sound NIZK to generate the proof component of an anamorphic ciphertext [16]. More recently, Krenn, Samelin, and Slamanig [12] advocated treating NIZK simulators as constructive components of cryptographic designs, including in the context of anamorphic encryption. Their work, however, does not formulate anamorphic NIZKs as a standalone primitive or characterize the additional security requirements that arise when the anamorphic functionality is built directly into a proof system.

Moving anamorphism from signatures to NIZKs also introduces a new security interface. A NIZK proof is required to reveal no information about the witness beyond what is implied by the statement. An anamorphic proof, however, must additionally support a second, secret-keyed interpretation available to the holder of the double key. The covert message recipient thus has a capability

that is absent from the standard NIZK model: it sees the proof together with the double key and can invoke a recovery procedure that is intentionally unavailable to other parties. Standard zero knowledge does not by itself characterize the security of this augmented view. In particular, enabling covert-message recovery should not give the recipient additional information about the witness beyond what is permitted by the underlying NIZK.

The same issue extends to stronger proof-system guarantees, such as simulation extractability. Once the double key gives the covert message recipient an additional capability for interpreting proofs, these guarantees must be reconsidered in the presence of that capability. This motivates a proof-system formulation that explicitly captures the recipient's additional power while preserving the underlying NIZK guarantees.

This raises a natural question:

> *Can anamorphism be realized directly in a NIZK while preserving the security guarantees of the underlying proof system?*

We answer this question by introducing *anamorphic non-interactive zero-knowledge proofs* (ANIZKs), in which a prover generates a publicly verifiable NIZK proof while embedding an additional message recoverable only with a double key. The same proof remains valid under the ordinary verification algorithm.

Main contributions. Our main contributions are as follows:

1. We introduce anamorphic non-interactive zero-knowledge (ANIZK) systems and formalize a security framework where covert embedding coexists with standard NIZK guarantees. Specifically, we define anamorphic security against detection and strong robustness against unauthorized proof forgery, while lifting zero-knowledge and simulation extractability to the anamorphic setting.
2. We provide a generic construction of ANIZK in the random oracle model from an authenticated encryption with associated data (AEAD) scheme and a Sigma protocol, using a variant of the Fiat-Shamir transform as the underlying technique. Beyond the standard Sigma protocol requirements, this construction requires only mild additional properties, which are either natively satisfied or attainable via simple modifications. Consequently, our approach applies to a broad class of Sigma protocols.
3. We demonstrate the applicability of ANIZK by deploying it as the underlying NIZK within the classic Naor-Yung paradigm. This straightforward integration yields a recipient anamorphic encryption (RAME) scheme achieving STE-xCCA1, SIND-aCCA1, and SIND-nCCA security, which constitute the maximal chosen-ciphertext (CCA) security profile formalized in [7] and collectively imply all other standard anamorphic security guarantees.[4]

[4] Note that unlike the RAME definition in [7], the anamorphic decryption algorithm in our RAME setting does not require the receiver's secret key as input. This structural difference allows an adversary holding the double key to perform local anamorphic

4. We extend our framework to asymmetric ANIZK (ASY-ANIZK), where an anamorphic public key enables covert embedding and an anamorphic secret key enables recovery. We construct ASY-ANIZK from a labeled NIZK and a public-key encryption scheme satisfying ciphertext pseudorandomness and random ciphertext rejection. Crucially, observing that the public nature of ASY-ANIZK inherently precludes strong robustness, we instead prove that our ASY-ANIZK scheme satisfies weak robustness.

ANIZK primitive. An ANIZK scheme for a relation $\mathcal{R}$ and an anamorphic message space $\mathcal{AM}$ has six algorithms: Setup, Prove, Verify, $\mathsf{aKeyGen}$, aProve, and $\mathsf{aExtract}$.

The first three algorithms provide the standard NIZK functionality. In particular, Setup generates a common reference string crs (which we leave implicit in the subsequent algorithms). Given a statement x and a witness w satisfying $(x, w) \in \mathcal{R}$, the prover generates $\pi \leftarrow \mathsf{Prove}(x, w)$, and anyone can efficiently verify the proof by running $\mathsf{Verify}(x, \pi)$.

The last three algorithms ($\mathsf{aKeyGen}, \mathsf{aProve}, \mathsf{aExtract}$) support covert communication. The algorithm $\mathsf{aKeyGen}$ generates a double key dk shared between the prover and the recipient. Given a statement x, a witness w (where $(x, w) \in \mathcal{R}$), and an anamorphic message $\mathsf{am} \in \mathcal{AM}$, the prover computes $\pi \leftarrow \mathsf{aProve}(x, w, \mathsf{dk}, \mathsf{am})$ to embed am while proving the statement x. Upon receiving π, the recipient invokes the anamorphic extraction algorithm $\mathsf{aExtract}(x, \pi, \mathsf{dk})$, obtaining either the embedded message or $\perp$. Anamorphic correctness requires that, for every honestly generated anamorphic proof, ordinary verifiers accept the proof and the recipient successfully recovers the embedded message.

Security requirements. Beyond the standard NIZK requirements (i.e., completeness, computational soundness, and zero knowledge), an ANIZK scheme must satisfy the following security properties.

1. *Anamorphic security.* This property ensures that the covert channel remains hidden even from a dictator who can force the prover to disclose the randomness associated with each proof. At its core, this security notion explicitly requires the existence of an efficient equivocation algorithm. For any anamorphic proof carrying a covert message, this algorithm must be able to produce alternative randomness that explains the proof as a legitimate output of the ordinary proving algorithm. Consequently, a dictator lacking the double key dk cannot distinguish an anamorphic proof paired with this alternative randomness from an ordinary proof accompanied by its true randomness.
2. *Strong robustness.* Strong robustness ensures that the capability to embed covert messages is strictly exclusive to the holder of the double key dk. Specifically, even after adaptively observing valid anamorphic proofs, an adversary lacking dk cannot forge any fresh proof, such that the extraction algorithm successfully recovers a covert message from the proof. As a corollary, this

decryptions, rendering certain security notions in [7] (e.g., SIND-xCCA1) inherently unachievable in our RAME setting.

property also ensures that ordinary proofs generated by the standard proving algorithm inherently carry no covert content.

3. *Recipient anamorphic zero knowledge.* It guarantees that an adversary holding dk learns nothing about the witness, apart from the covert message and the fact that the statement is true. As with the standard zero-knowledge property, we define recipient anamorphic zero knowledge via simulation: given a true statement and a covert message, a simulator must produce an indistinguishable anamorphic proof without using the witness while keeping the covert message recoverable by the anamorphic extraction algorithm.
4. *Recipient anamorphic simulation extractability.* Recipient anamorphic simulation extractability ensures that, even after adaptively obtaining simulated anamorphic proofs, an adversary possessing the double key cannot forge a valid proof without knowing the underlying witness. At its core, this security notion explicitly requires the existence of an efficient extractor capable of successfully recovering a valid witness from any fresh accepting proof output by the adversary. As a corollary, this property inherently implies recipient anamorphic simulation soundness, ruling out the creation of accepting proofs for false statements in the same setting.

ANIZK framework. Here, we show the ANIZK framework. Before that, we firstly recall some building blocks.

Building blocks. The construction uses three building blocks. *(i) A Sigma protocol* $\mathsf{Sigma} = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ for $\mathcal{R}$. Besides the normal security properties, it additionally requires *quasi-unique responses property* and *commitment unpredictability. (ii) An authenticated encryption with associated data (AEAD) scheme* $\mathsf{AEAD} = (\mathsf{AEAD.Enc}, \mathsf{AEAD.Dec})$ with key space $\{0,1\}^\lambda$ and ciphertext space $\{0,1\}^{\ell(\lambda)}$. It should satisfy IND$-CPA and INT-CTXT security. IND$-CPA security requires pseudorandom ciphertexts, while INT-CTXT guarantees that without the secret key, no adversary can forge a valid ciphertext that decrypts to a non-empty message. *(iii) A hash family* $\mathcal{H}$, whose member H maps strings to the challenge space of the Sigma protocol and is modeled as a random oracle.

Main idea. Our construction utilizes a variant of Fiat-Sharmir transform as the underlying technique, called a *salted Fiat-Shamir transform.* In the ordinary Fiat-Shamir transform, the challenge is derived by hashing the statement and the prover's first message. We simply append one extra parameter, a *salt* S, that is carried inside the proof and included in the hash input, yet is otherwise invisible to the underlying Sigma protocol. The ordinary and anamorphic modes differ only in how this salt is produced. Ordinary proving draws S uniformly from $\{0,1\}^{\ell(\lambda)}$, whereas anamorphic proving sets

$$S \leftarrow \mathsf{AEAD.Enc}(\mathsf{dk}, \mathsf{am}, \mathsf{ad} = (x, a)),$$

so that the salt is an authenticated encryption of the covert message am under the double key dk, bound to the particular statement and commitment (x, a). Since AEAD is IND$-CPA secure, i.e., the ciphertext is pseudorandom, an observer

without dk cannot distinguish the two modes. In addition, as AEAD is INT-CTXT secure the ciphertext is bound to (x, a), a valid covert payload cannot be transplanted into another proof. More importantly, recovering the salt leaks nothing about the witness.

Detailed construction. Our generic scheme $\mathsf{ANIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{aKeyGen}, \mathsf{aProve}, \mathsf{aExtract})$ for relation $\mathcal{R}$ is constructed as follows.

The setup algorithm Setup samples a hash function $\mathsf{H} \leftarrow \mathcal{H}$. On input $(x, w) \in \mathcal{R}$, the proving algorithm Prove computes the commitment of the Sigma protocol $(a, \mathsf{st}) \leftarrow \mathsf{P}_1(x, w)$, samples $S \leftarrow \{0,1\}^{\ell(\lambda)}$, and then derives the challenge and response similarly to an ordinary Fiat-Shamir proof,

$$c = \mathsf{H}(x, a, S), \qquad z \leftarrow \mathsf{P}_2(a, c, x, w, \mathsf{st}),$$

returning $\pi = (a, S, z)$. Verification algorithm Verify re-computes c and invokes the verification algorithm of the Sigma protocol to check the proof, i.e., $\mathsf{V}_2(x, a, c, z)$.

The anamorphic algorithms mirror this. The key generation algorithm $\mathsf{aKeyGen}$ returns a uniform key $\mathsf{dk} \leftarrow \{0,1\}^{\lambda}$, then the anamorphic proving algorithm aProve runs the same Sigma protocol on the same (x, w) but substitutes the uniform salt with the AEAD encryption $S \leftarrow \mathsf{AEAD.Enc}(\mathsf{dk}, \mathsf{am}, \mathsf{ad} = (x, a))$. Finally, $\mathsf{aExtract}$ parses $\pi = (a, S, z)$ and returns $\mathsf{AEAD.Dec}(\mathsf{dk}, S, \mathsf{ad} = (x, a))$, i.e., the message or $\perp$.

Ordinary and anamorphic proofs share the same format, length, and public verification. Completeness and anamorphic correctness directly follow from the correctness of AEAD and the completeness of the Sigma protocol.

Security analysis for ANIZK. We briefly analyze the security of ANIZK.

Ordinary NIZK security. The ordinary salt is independent of the witness and the hash function is modeled as a random oracle. Thus, the ordinary NIZK security properties are directly guaranteed by the Fiat-Shamir transform.

Anamorphic security. For an anamorphic proof $\pi = (a, S, z)$, the equivocation algorithm can simply output $r_{\mathsf{fake}} = (r, S)$ where r is the inner randomness used by the prover of the underlying Sigma protocol. Running the ordinary proving algorithm Prove with this randomness $r_{\mathsf{fake}} = (r, S)$ would reproduce the same commitment and response (the challenge is fixed $c = \mathsf{H}(x, a, S)$). In this way, the ciphertext S can be viewed as a salt in ordinary proving algorithm, while the covert message remains undisclosed. In addition, the IND\$-CPA security of AEAD guarantees that the ciphertext S is indistinguishable from a string uniformly sampled over $\{0,1\}^{\ell(\lambda)}$, which implies the indistinguishability of ordinary proof and anamorphic proof. Thus, ANIZK achieves the anamorphic security.

Strong robustness. Strong robustness is mainly guaranteed by the INT-CTXT security of AEAD and the quasi-unique responses property of the underlying Sigma protocol. The winning condition of strong robustness is that the adversary outputs a fresh accepting pair (x^*, π^*), where $\pi^* = (a^*, S^*, z^*)$, such that $\mathsf{aExtract}$ can recover a non-empty message from π^*. We then consider the pair $(S^*, (x^*, a^*))$. (i) If this pair is new, i.e., $(S^*, (x^*, a^*))$ never appears among the

previous queries, then it constitutes a valid forgery for the INT-CTXT game of AEAD. (ii) Otherwise, $(S^*, (x^*, a^*))$ appears in some previous anamorphic proof query. In other words, the adversary has reused a previously queried salt for the same statement and commitment. This implies that the challenge derived via $\mathsf{H}(x^*, a^*, S^*)$ is the same as the one used in the previous query. Due to the quasi-unique responses property of the underlying Sigma protocol, the prover must output the same response as in the previous query. Therefore, $\pi^* = (a^*, S^*, z^*)$ is identical to the proof in the previous query, which contradicts the winning condition of strong robustness.

Recipient anamorphic zero knowledge. We firstly construct a simulator in the following way. Given a statement and a covert message (x, am), the simulator firstly runs the special HVZK simulator of the underlying Sigma protocol with a random challenge c to generate an accepting transcript (a, c, z). Then, it invokes AEAD to generate an encryption of am, $S \leftarrow \mathsf{AEAD.Enc}(\mathsf{dk}, \mathsf{am}, \mathsf{ad} = (x, a))$, using double key dk. Finally, it programs $\mathsf{H}(x, a, S)$ to match the challenge c, and outputs $\pi = (a, S, z)$ as the anamorphic proof for the statement x. Then, the proof for recipient anamorphic zero knowledge is summarized as follows. Firstly, due to the commitment unpredictability, it is with only negligible probability to distinguish explicitly programming the random oracle during proof generation from lazy sampling. Secondly, the special HVZK guarantees that the simulated transcript is indistinguishable from real one.

Recipient anamorphic simulation extractability. Given a PPT adversary, the extractor has black-box access to the adversary and maintains its full transcript. For any adversarially produced fresh and accepting statement-proof pair $(x^*, \pi^* = (a^*, S^*, z^*))$, the extractor firstly checks whether the challenge $c^* = \mathsf{H}(x^*, a^*, S^*)$ is programmed by the simulation oracle. If the challenge was programmed by the simulator, two distinct responses for the same commitment and challenge would violate the quasi-unique responses property of the underlying Sigma protocol. Otherwise, special soundness of the Sigma protocol guarantees that the forked transcripts yield a valid witness, contradicting the winning conditions of recipient anamorphic simulation extractability.

Application to recipient anamorphic encryption. We illustrate the application of ANIZK by constructing a recipient anamorphic encryption (RAME) scheme.

An RAME scheme consists of six algorithms $\mathsf{RAME} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{aKeyGen}, \mathsf{aEnc}, \mathsf{aDec})$, where the first three algorithms constitute a standard public-key encryption (PKE) scheme, and the latter three form an anamorphic triplet for covert communications. Specifically, $\mathsf{aKeyGen}$ generates a double key dk alongside the standard key pair $(\mathsf{pk}, \mathsf{sk})$. In the anamorphic mode, a sender uses dk to run aEnc, embedding a covert message into an anamorphic ciphertext. It is required that this ciphertext decrypts correctly to the normal message under sk, while a recipient holding dk can extract the covert message via aDec.

At a high level, our generic RAME scheme RAME is built upon the classic Naor-Yung paradigm [15]. The building blocks include a PKE scheme $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ and an ANIZK scheme ANIZK for a relation

$\mathcal{R}_{\mathsf{NY}}$ in Eq. (3) establishing that two PKE ciphertexts encrypt the same message. The overall public key pk incorporates two independently generated PKE public keys, pk_1 and pk_2. Let $\mathsf{sk}_1, \mathsf{sk}_2$ be the secret keys corresponding to the public keys $\mathsf{pk}_1, \mathsf{pk}_2$, respectively. The overall secret key sk is $(\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{sk}_1)$. The double key dk of RAME is set to be that of the underlying ANIZK.

- *In the normal encryption mode:* The Enc algorithm of RAME encrypts a message m as follows. It first computes two ciphertexts, ct_1 and ct_2, via PKE.Enc under pk_1 and pk_2, respectively. It then invokes ANIZK.Prove to generate an NIZK proof π demonstrating that both ct_1 and ct_2 encrypt the same underlying message. Finally, it outputs the ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$.
- *In the anamorphic encryption mode:* The aEnc algorithm of RAME encrypts a normal message m and embeds a covert message am as follows. It first encrypts m via PKE.Enc under pk_1 and pk_2, obtaining two ciphertexts ct_1 and ct_2, respectively. It then invokes ANIZK.aProve with the double key dk to generate an anamorphic proof π, which embeds am while still demonstrating that both ct_1 and ct_2 encrypt the same underlying message m. Finally, it outputs the anamorphic ciphertext $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$.
- *In the normal decryption mode:* The Dec algorithm of RAME decrypts a ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ as follows. It first invokes ANIZK.Verify to check the validity of the proof π. If the verification fails, it outputs $\perp$. Otherwise, it decrypts ct_1 via PKE.Dec using sk_1, and outputs the resulting message.
- *In the anamorphic decryption mode:* The aDec algorithm of RAME extracts an anamorphic message from an anamorphic ciphertext $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ as follows. It first invokes ANIZK.Verify to check the validity of the proof π. If the verification fails, it outputs $\perp$. Otherwise, it extracts the anamorphic message am from π via ANIZK.aExtract using dk, and outputs it.

Security Analysis. We limit our security analysis of RAME to STE-xCCA1, SIND-aCCA1, and SIND-nCCA. These properties are as defined by Choi et al. [7]. Within their framework, these notions constitute the strongest jointly achievable security combination compatible with our symmetric anamorphic encryption model, in which anamorphic decryption requires only the public key and the double key.

Informally, STE-xCCA1 ensures that a dictator holding pk and sk cannot distinguish normal operation from anamorphic operation, even with access to an anamorphic decryption oracle. This oracle always returns $\perp$ in normal mode and provides anamorphic decryptions of non-challenge ciphertexts in anamorphic mode. STE-xCCA1 security of RAME can be reduced to the anamorphic security and strong robustness of ANIZK. Specifically, the $(\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{sk}_1, \mathsf{sk}_2)$ have the same distribution in both modes, and two PKE ciphertext components are generated identically. Consider the challenging oracle between two modes, the anamorphic security of ANIZK allows the normal proofs to be replaced with anamorphic proofs in the dictator's view. We then modify the anamorphic decryption oracle in normal mode to behave as it does in anamorphic mode. If the dictator can distinguish this change, there must be a decryption query on a ciphertext $\mathsf{act}^* = (\mathsf{ct}_1^*, \mathsf{ct}_2^*, \pi^*)$ that was not returned by the challenge oracle and

whose proof component π^* allows $\mathsf{ANIZK.aExtract}$ to recover a valid anamorphic message. So it is contradicted by the strong robustness of ANIZK.

Basically, SIND-aCCA1 requires that an adversary holding pk and dk cannot distinguish whether an anamorphic ciphertext encrypts the ordinary message m_0 or m_1, even with access to ordinary decryptions of non-challenge ciphertexts. Note that in our construction, each challenge ciphertext has the form $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$, where both PKE components encrypt the same ordinary message and π embeds the chosen anamorphic message. For π, the recipient anamorphic zero knowledge of ANIZK allows us to replace it with a simulated proof carrying the same anamorphic message, without noticeably changing the adversary's view despite its knowledge of dk. For any non-challenge ciphertext $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ submitted to the decryption oracle, if π is valid, the recipient anamorphic simulation soundness guarantees $x = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{pk}_1, \mathsf{pk}_2) \notin \mathcal{L}_{\mathcal{R}_{\mathsf{NY}}}$, except with negligible probability. Hence, the two PKE components encrypt the same ordinary message. Thus, decrypting either component gives the same response. We can therefore change the challenge plaintext from m_0 to m_1 one component at a time, while answering decryption queries using the other component. Any noticeable distinction between these changes would contradict the IND-CPA security of PKE. Finally, recipient anamorphic zero knowledge allows us to restore honest anamorphic proofs, establishing SIND-aCCA1 security.

Informally, SIND-nCCA requires that an adversary holding pk and dk cannot distinguish whether a normal ciphertext encrypts m_0 or m_1, even with access to ordinary decryptions of non-challenge ciphertexts. For this guarantee, we additionally require the ordinary NIZK used in RAME, formed by the algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ of ANIZK, to satisfy simulation soundness (Definition 18). The proof follows a similar approach to that of SIND-aCCA1 security and is omitted due to space constraints.

Asymmetric ANIZK. In the setting described above, the prover and the recipient share a double key that enables both message embedding and recovery. To separate these capabilities, we introduce asymmetric anamorphic NIZK (ASY-ANIZK).

An ASY-ANIZK scheme consists of six algorithms ($\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{aKeyGen}, \mathsf{aProve}, \mathsf{aExtract}$). Compared with ANIZK, the ordinary algorithms Setup, Prove, and Verify retain the same interfaces, while the key interfaces of the three anamorphic algorithms are modified. Specifically, $\mathsf{aKeyGen}$ generates an anamorphic key pair $(\mathsf{apk}, \mathsf{ask})$ instead of a shared double key dk. The algorithm aProve uses the anamorphic public key apk to embed a message, while $\mathsf{aExtract}$ uses the anamorphic secret key ask to recover it. Multiple provers with valid witnesses can therefore send anamorphic messages to the same recipient without sharing its secret extraction key.

ASY-ANIZK retains the ordinary NIZK guarantees and admits analogous notions of anamorphic security considered in the symmetric setting, including recipient anamorphic zero knowledge, and recipient anamorphic simulation extractability. However, since the embedding key apk is public, strong robustness is inherently unattainable: anyone with apk and a valid witness can generate a

fresh accepting proof that carries a recoverable anamorphic message. The appropriate robustness guarantee for ASY-ANIZK is therefore *weak robustness*, which requires that an honestly generated ordinary proof yields $\perp$ under anamorphic extraction, except with negligible probability.

ASY-ANIZK construction. The construction relies on a primitive called labeled NIZK (LNIZK). A LNIZK scheme $\mathsf{LNIZK} = (\mathsf{LNIZK.Setup}, \mathsf{LNIZK.Prove}, \mathsf{LNIZK.Verify})$ for a relation $\mathcal{R}$, with label space $\mathcal{LS}$, is similar to an ordinary NIZK, except that both the proving and verification algorithms additionally take a label as input. The role of the label is to bind a proof to specific associated data, which is a well-established technique to achieve non-malleability and simulation extractability [4]. Actually, the security notions of LNIZK are similar to those of ordinary NIZK, as defined in Definition 31. The construction also employs a PKE scheme $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ with ciphertext space $\mathcal{CT} = \mathcal{LS}$. Similar to the ANIZK framework, the PKE is required to be IND$-CPA security. We additionally require that the PKE supports random ciphertext rejection: a uniformly random string $\mathsf{ct} \leftarrow \mathcal{CT}$ decrypts to a valid message with only negligible probability.

Then, we outline the generic construction of ASY-ANIZK $\mathsf{ASY\text{-}ANIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{aKeyGen}, \mathsf{aProve}, \mathsf{aExtract})$ from LNIZK and PKE.

The setup algorithm Setup invokes $\mathsf{LNIZK.Setup}$ and returns the resulting common reference string crs. The anamorphic key generation algorithm $\mathsf{aKeyGen}$ invokes $\mathsf{PKE.KeyGen}$ to generate the key pair $(\mathsf{apk}, \mathsf{ask})$.

Given a valid statement–witness pair (x, w), the ordinary proving algorithm Prove samples a label ρ uniformly from $\mathcal{LS}$, generates $\pi' \leftarrow \mathsf{LNIZK.Prove}(x, w, \rho)$, and returns the proof $\pi = (\pi', \rho)$. Verification is straightforward: Verify parses $\pi = (\pi', \rho)$ and outputs the result of $\mathsf{LNIZK.Verify}(x, \pi', \rho)$.

The anamorphic proving algorithm aProve additionally takes as input the public embedding key apk and an anamorphic message am. It mirrors the ordinary proving algorithm, except that it computes a ciphertext $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{apk}, \mathsf{am})$ and uses this ciphertext as the label of the underlying LNIZK proof. To recover the anamorphic message, the extraction algorithm $\mathsf{aExtract}$ parses $\pi^* = (\pi', \mathsf{ct})$ and outputs $\mathsf{PKE.Dec}(\mathsf{ask}, \mathsf{ct})$.

The completeness and anamorphic correctness of $\mathsf{ASY\text{-}ANIZK}$ follows immediately from the completeness of LNIZK and the decryption correctness of PKE.

Security Analysis for ASY-ANIZK. The construction of ASY-ANIZK closely parallels that of ANIZK: covert messages are embedded into *labels* in ASY-ANIZK, just as they are embedded into *salts* in ANIZK. Consequently, the proofs of most security properties of ASY-ANIZK follow analogously to those of ANIZK, and here we only briefly analyze the two properties that deserve particular attention: weak robustness and recipient anamorphic simulation extractability.

Since strong robustness is inherently unattainable for ASY-ANIZK, weak robustness serves as the necessary security relaxation. Recipient anamorphic simulation extractability, in turn, was proved for ANIZK relying on the quasi-unique responses property of the underlying Sigma protocol, a property that is

not supported by the building blocks of ASY-ANIZK. Its proof therefore requires a different approach.

Weak robustness. Weak robustness follows directly from the random ciphertext rejection property of PKE. The label ρ used by ordinary proving algorithm is sampled uniformly from $\mathcal{CT}$ and independently of the adversary. Random ciphertext rejection property guarantees that with overwhelming probability it does not decrypt to a valid message under ask.

Recipient anamorphic simulation extractability. This property is ensured by the simulation extractability of the underlying LNIZK. Let (x^*, π^*) be a fresh accepting pair output by the adversary, where $\pi^* = (\pi'^*, \mathsf{ct}^*)$. Since the verification algorithm of ASY-ANIZK accepts it, π'^* is an accepting labeled proof for x^* under label ct^*. Then, it implies that this tuple $(x^*, \mathsf{ct}^*, \pi'^*)$ is also fresh for the LNIZK simulator as well. Simulation extractability of LNIZK then yields a valid witness for x^*, except with negligible probability, which contradicts the winning conditions of recipient anamorphic simualtion extractability.

Discussions. The following discussion outlines the potential applications of ANIZK and ASY-ANIZK, the scope of their security guarantees, and the issue of auxiliary parameters.

Potential applications. ANIZK enables a prover to establish the validity of a public statement while embedding a covert message for a designated recipient. Our RAME scheme serves as a specific application of this primitive in the encryption setting. Beyond this, the utility of ANIZK and ASY-ANIZK extends more broadly. Specifically, they could potentially facilitate the design of anamorphic signatures and authenticated key-exchange protocols that employ NIZK proofs. The formal constructions and rigorous security analyses of these extensions are left for future work.

Security scope. Our security notions capture post-execution coercion scenarios where a dictator intercepts a proof. Specifically, the dictator demands the underlying randomness from the prover. Provided the anamorphic keys (dk or ask) remain secret, the coerced party can successfully equivocate, explaining the proof as a normal one without covert messages. As is standard in anamorphic cryptography, this guarantee assumes the adversary cannot compromise the transient internal state during algorithm execution.

On the auxiliary parameter. Our constructions require an additional parameter in the ordinary proof format, a salt or label, which the anamorphic prover repurposes as the covert channel. Avoiding this additional parameter altogether and keeping the proof format unchanged would be preferable. We leave this as an open problem.

Related work. Persiano, Phan, and Yung [16] introduced anamorphic encryption to support covert communication despite exposure of ordinary decryption keys and messages. Banfi et al. [2] refined the model and studied robustness, while Choi et al. [7] systematized security notions and their relationships, including chosen-ciphertext security. Other developments include asymmetric and public-key anamorphism [6,17] and multi-message extensions [1]. These works

provide the encryption setting and security terminology for our RAME application. ASY-ANIZK studies the separation of public embedding and secret recovery within a proof system.

Kutyłowski et al. [13] introduced anamorphic signatures and discussed obtaining them from anamorphic protocols through the Fiat-Shamir transform. Jaeger and Stracovsky [11] strengthened unforgeability definitions to address malicious dictators and recipients, and Deo and Libert [8] developed constructions supporting long anamorphic messages with both guarantees. These works motivate treating the recipient as a separate adversary. Their unforgeability guarantees, however, do not by themselves establish witness privacy or witness extraction after access to simulated proofs. Our framework studies these additional requirements when the carrier is a NIZK proof, preserving the distinction between permission to recover covert messages and access to the underlying witness.

Roadmap. The paper is organized as follows. Section 2 reviews the preliminaries. Section 3 then defines ANIZK and its security notions. We present the ANIZK construction in Section 4 and develop the RAME application in Section 5. Section 6 introduces ASY-ANIZK and its construction. Finally, additional definitions, technical details, and security proofs are provided in the appendices.

2 Preliminaries

Notations. Let λ be the security parameter, and $\mathsf{negl}(\lambda)$ be a negligible function in λ. Let $[k] := \{1, \ldots, k\}$ for $k \in \mathbb{N}$, and $a \leftarrow \mathsf{Set}$ (resp. Dist) denote uniform sampling from a finite set Set (resp. sampling from a distribution Dist). For a PPT algorithm Alg, let $\mathcal{RS}_{\mathsf{Alg}}$ be its randomness space; $y \leftarrow \mathsf{Alg}(x)$ denotes execution with uniform randomness, and $\mathsf{Alg}(x; r)$ execution with $r \in \mathcal{RS}_{\mathsf{Alg}}$. We also use $\leftarrow$ for assignment.

A *polynomial-time* relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ has membership of $(x, w) \in \mathcal{R}$ decidable in time polynomial in $|x|$, where $\mathcal{X}$ and $\mathcal{W}$ are the statement and witness spaces. If $(x, w) \in \mathcal{R}$, then w is a *witness* for statement x. The associated NP language is $\mathcal{L}_{\mathcal{R}} = \{x \in \mathcal{X} \mid \exists w \in \mathcal{W} : (x, w) \in \mathcal{R}\}$.

Authenticated encryption with associated data (AEAD). AEAD is a symmetric-key primitive providing confidentiality and integrity for messages and authenticity of associated data. We use the following syntax [19,18].

Definition 1 (AEAD). *An AEAD scheme* AEAD*, with key space* $\mathcal{K}$*, message space* $\mathcal{M}$*, associated data space* $\mathcal{AD}$*, and ciphertext space* $\mathcal{CT}$*, consists of the following PPT algorithms:*

- $\mathsf{Enc}(\mathsf{key}, \mathsf{m}, \mathsf{ad}) \to \mathsf{ct}$*: The encryption algorithm takes a symmetric key* $\mathsf{key} \in \mathcal{K}$*, a message* $\mathsf{m} \in \mathcal{M}$*, and associated data* $\mathsf{ad} \in \mathcal{AD}$ *as input, and then outputs a ciphertext* $\mathsf{ct} \in \mathcal{CT}$*.*
- $\mathsf{Dec}(\mathsf{key}, \mathsf{ct}, \mathsf{ad}) \to \mathsf{m}/\perp$*: The decryption takes a symmetric key* $\mathsf{key} \in \mathcal{K}$*, a ciphertext* $\mathsf{ct} \in \mathcal{CT}$*, and associated data* $\mathsf{ad} \in \mathcal{AD}$ *as input, and then outputs* $\mathsf{m} \in \mathcal{M}$ *or a special symbol* $\perp$ *indicating an invalid or unauthentic ciphertext.*

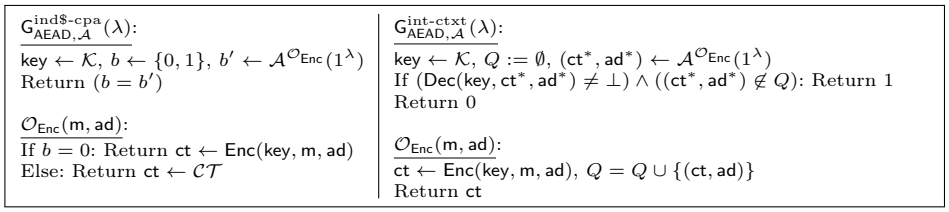

Fig. 1: IND$-CPA and INT-CTXT security games for AEAD

Correctness requires that for any $\mathsf{key} \in \mathcal{K}$*, any* $\mathsf{m} \in \mathcal{M}$*, and any* $\mathsf{ad} \in \mathcal{AD}$*,* $\mathsf{Dec}(\mathsf{key}, \mathsf{Enc}(\mathsf{key}, \mathsf{m}, \mathsf{ad}), \mathsf{ad}) = \mathsf{m}$.

We recall the IND$-CPA and INT-CTXT security definitions for AEAD.

Definition 2 (IND$-CPA Security). *An AEAD scheme* AEAD *is IND$-CPA secure, if for any PPT adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathsf{AEAD},\mathcal{A}}(\lambda) := |\Pr[\mathsf{G}^{\text{ind\$-cpa}}_{\mathsf{AEAD},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\text{ind\$-cpa}}_{\mathsf{AEAD},\mathcal{A}}(\lambda)$ *is defined in Fig. 1.*

Definition 3 (INT-CTXT Security). *An AEAD scheme* AEAD *is INT-CTXT secure, if for any PPT adversary* $\mathcal{A}$*,*

$$Adv^{\text{int-ctxt}}_{\mathsf{AEAD},\mathcal{A}}(\lambda) = \Pr[\mathsf{G}^{\text{int-ctxt}}_{\mathsf{AEAD},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\text{int-ctxt}}_{\mathsf{AEAD},\mathcal{A}}(\lambda)$ *is defined in Fig. 1.*

Non-interactive zero knowledge system (NIZK). Now we recall the definition of NIZK system.[5]

Definition 4 (NIZK). *An NIZK scheme for* $\mathcal{R}$ *consists of three PPT algorithms* $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *defined as follows:*

- $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{crs}$*: The setup algorithm takes the security parameter* λ *as input, and then outputs a common reference string* crs*, which is implicitly passed to all subsequent algorithms and omitted for brevity.*
- $\mathsf{Prove}(x, w) \rightarrow \pi$*: The proving algorithm takes a statement* x *and a witness* w*, where* $(x, w) \in \mathcal{R}$*, as input, and then outputs a proof* π*.*
- $\mathsf{Verify}(x, \pi) \rightarrow b$*: The verification algorithm takes a statement* x *and a proof* π *as input, and then outputs* $b = 1$ *if it accepts the proof* π*; otherwise, it outputs* $b = 0$*.*

[5] We use "NIZK system" as a generic term; under the standard proof/argument distinction, computational soundness corresponds to an NIZK argument. We also occasionally use the term "NIZK proof", following its use in some related works [14,5].

$\underline{\mathsf{G}^{\text{zk-real}}_{\mathsf{NIZK},\mathcal{A}}(\lambda)}$:	$\underline{\mathsf{G}^{\text{zk-sim}}_{\mathsf{NIZK},\mathcal{A},\mathcal{S}}(\lambda)}$:
$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$	$(\mathsf{crs}, \tau) \leftarrow \mathcal{S}.\mathsf{Setup}(1^\lambda)$
Return $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Prove}}}(\mathsf{crs})$	Return $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sim}}}(\mathsf{crs})$
$\underline{\mathcal{O}_{\mathsf{Prove}}(x, w)}$:	$\underline{\mathcal{O}_{\mathsf{Sim}}(x, w)}$:
If $(x, w) \notin \mathcal{R}$: Return $\perp$	If $(x, w) \notin \mathcal{R}$: Return $\perp$
Return $\mathsf{Prove}(x, w)$	Return $\mathcal{S}.\mathsf{Prove}(\tau, x)$

Fig. 2: Zero knowledge game for NIZK

We require the following properties.
Completeness. For all $(x, w) \in \mathcal{R}$,

$$\Pr[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \pi \leftarrow \mathsf{Prove}(x, w) : \mathsf{Verify}(x, \pi) = 1] \geq 1 - \mathsf{negl}(\lambda).$$

Computational Soundness. For any PPT adversary $\mathcal{A}$*, it holds that*

$$\Pr[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) : \mathsf{Verify}(x, \pi) = 1 \wedge x \notin \mathcal{L}_\mathcal{R}] \leq \mathsf{negl}(\lambda).$$

Zero Knowledge. There exists a PPT simulator $\mathcal{S} = (\mathcal{S}.\mathsf{Setup}, \mathcal{S}.\mathsf{Prove})$*, such that for any PPT adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\text{zk}}_{\mathsf{NIZK},\mathcal{A},\mathcal{S}}(\lambda) = |\Pr[\mathsf{G}^{\text{zk-real}}_{\mathsf{NIZK},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{G}^{\text{zk-sim}}_{\mathsf{NIZK},\mathcal{A},\mathcal{S}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{G}^{\text{zk-real}}_{\mathsf{NIZK},\mathcal{A}}(\lambda)$ *and* $\mathsf{G}^{\text{zk-sim}}_{\mathsf{NIZK},\mathcal{A},\mathcal{S}}(\lambda)$ *can be found in Fig. 2.*

Sigma Protocol. A Sigma protocol $\mathsf{Sigma} = (\mathsf{P}, \mathsf{V})$ consists of a prover $\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2)$ and a verifier $\mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2)$. For any $(x, w) \in \mathcal{R}$, $\mathsf{P}(x, w)$ first sends to V a commitment a computed via $(a, \mathsf{st}) \leftarrow \mathsf{P}_1(x, w)$;[6] then V_1 returns a random challenge $c \in \mathcal{CL}$, to which P replies with $z \leftarrow \mathsf{P}_2(a, c, x, w, \mathsf{st})$; finally V_2 outputs $b \in \{0, 1\}$. We require that V makes no random choices other than selecting c. The tuple (a, c, z) is called a *conversation/transcript*, and is *accepting* for x if the final output of $\mathsf{V}(x)$ is 1. Completeness requires that for all $(x, w) \in \mathcal{R}$, when $\mathsf{P}(x, w)$ and $\mathsf{V}(x)$ interact, the final output of $\mathsf{V}(x)$ is always 1.

Security notions for Sigma protocols. The corresponding security notions for Sigma protocols are recalled as follows.

Definition 5 (Soundness). *If* $x \notin \mathcal{L}_\mathcal{R}$*, then any malicious (even unbounded) prover* P^* *cannot output an accepting transcript (interacting with* V*) except with negligible probability.*

Definition 6 (Special Soundness). *A Sigma protocol* $\mathsf{Sigma} = (\mathsf{P}, \mathsf{V})$ *for* $\mathcal{R}$ *provides* special soundness*, if there is an efficient deterministic algorithm* Ext *such that on input* $x \in \mathcal{X}$ *and two accepting conversations* $(a, c, z), (a, c', z')$ *where* $c \neq c'$*,* Ext *always outputs a witness* $w \in \mathcal{W}$ *satisfying* $(x, w) \in \mathcal{R}$*.*

[6] Sometimes we directly write $a \leftarrow \mathsf{P}_1(x, w)$ without explicitly writing st.

Definition 7 (Special HVZK). *A Sigma protocol* $\mathsf{Sigma} = (\mathsf{P}, \mathsf{V})$ *for* $\mathcal{R}$ *is special HVZK, if there exists a PPT simulator* $\mathcal{S}$ *such that for any adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,*

$$\mathbf{Adv}^{\text{shvzk}}_{\mathsf{Sigma},\mathcal{A},\mathcal{S}}(\lambda) = |\Pr[\mathsf{G}^{\text{shvzk-real}}_{\mathsf{Sigma},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{G}^{\text{shvzk-sim}}_{\mathsf{Sigma},\mathcal{A},\mathcal{S}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\text{shvzk-real}}_{\mathsf{Sigma},\mathcal{A}}(\lambda)$ *and* $\mathsf{G}^{\text{shvzk-sim}}_{\mathsf{Sigma},\mathcal{A},\mathcal{S}}(\lambda)$ *are defined in Fig. 3.*

$\underline{\mathsf{G}^{\text{shvzk-real}}_{\mathsf{Sigma},\mathcal{A}}(\lambda)}$:	$\underline{\mathsf{G}^{\text{shvzk-sim}}_{\mathsf{Sigma},\mathcal{A},\mathcal{S}}(\lambda)}$:
$(x, w, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda)$	$(x, w, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda)$
If $(x, w) \notin \mathcal{R}$: Return 0	If $(x, w) \notin \mathcal{R}$: Return 0
$(a, \mathsf{st}_p) \leftarrow \mathsf{P}_1(x, w)$, $c \leftarrow \mathsf{V}_1(a)$, $z \leftarrow \mathsf{P}_2(a, c, x, w, \mathsf{st}_p)$	$c \leftarrow \mathcal{CL}$, $(a, z) \leftarrow \mathcal{S}(x, c)$
Return $b \leftarrow \mathcal{A}_2(\mathsf{st}, a, c, z)$	Return $b \leftarrow \mathcal{A}_2(\mathsf{st}, a, c, z)$

Fig. 3: Special-HVZK game for Sigma protocol

A non-standard condition that many Sigma protocols satisfy requires that responses are quasi-unique, i.e., given an accepting proof it should be infeasible to find a new valid response for that proof.

Definition 8 (Quasi-unique responses). *A Sigma protocol* $\mathsf{Sigma} = (\mathsf{P}, \mathsf{V})$ *for* $\mathcal{R}$ *has quasi-unique responses if for any PPT adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\text{qur-uni}}_{\mathsf{Sigma},\mathcal{A}}(\lambda) = \Pr\Big[(x, a, c, z, z') \leftarrow \mathcal{A}(\lambda) : (\mathsf{V}_2(x, a, c, z) = \mathsf{V}_2(x, a, c, z') = 1) \wedge (z \neq z')\Big] \leq \mathsf{negl}(\lambda).$$

A Sigma protocol has unique responses if the probability above is zero.

Definition 9 (Commitment unpredictability). *A Sigma protocol* $\mathsf{Sigma} = (\mathsf{P}, \mathsf{V})$ *for* $\mathcal{R}$ *has commitment unpredictability, if for all* $(x, w) \in \mathcal{R}$*, it holds that* $\max_{a^*} \Pr[a \leftarrow \mathsf{P}_1(x, w) : a = a^*] \leq 2^{-\mu(\lambda)}$ *where* $\mu(\lambda) = \omega(\log \lambda)$*.*

Due to space constraints, we defer the notion of simulation soundness for NIZK, the definition of the Fiat-Shamir heuristic, the general forking lemma, and the definition and security of public key encryption to Appendix A.

3 Anamorphic NIZK Definition

In this section, we introduce a primitive called *anamorphic non-interactive zero-knowledge proof system (ANIZK)*, and formalize its security definitions. Generally speaking, an ANIZK scheme augments a conventional NIZK system by embedding a covert communication channel. In addition to generating ordinary proofs, a prover can use a double key to embed an anamorphic message into a valid proof, while a recipient holding the same double key can recover the embedded message.

3.1 Definition of ANIZK

We introduce the definition of ANIZK as follows.

Definition 10 (ANIZK). *An anamorphic non-interactive zero-knowledge proof system* $\mathsf{ANIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{aKeyGen}, \mathsf{aProve}, \mathsf{aExtract})$ *for relation* $\mathcal{R}$ *and anamorphic message space* $\mathcal{AM}$ *consists of a standard NIZK proof system* $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *(as defined in Definition 4), along with three additional algorithms* $(\mathsf{aKeyGen}, \mathsf{aProve}, \mathsf{aExtract})$ *defined as follows:*[7]

- $\mathsf{aKeyGen}(1^\lambda) \to \mathsf{dk}$*: The anamorphic key generation algorithm takes the security parameter* λ *as input, and then outputs a double key* dk*.*
- $\mathsf{aProve}(x, w, \mathsf{dk}, \mathsf{am}) \to \pi^*$*: The anamorphic proving algorithm takes as input a statement* x*, a witness* w*, a double key* dk *and an anamorphic message* am*, where* $(x, w) \in \mathcal{R}$*, and then outputs a proof* π^**.*
- $\mathsf{aExtract}(x, \pi^*, \mathsf{dk}) \to \mathsf{am}/\perp$*: The anamorphic extraction algorithm takes a statement* x*, a proof* π^* *and a double key* dk *as input, and then outputs an anamorphic message* am *or a failure symbol* $\perp$*.*

Anamorphic correctness requires that for any $\mathsf{am} \in \mathcal{AM}$ *and any* $(x, w) \in \mathcal{R}$*,*

$$\Pr\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{dk} \leftarrow \mathsf{aKeyGen}(1^\lambda) \\ \pi^* \leftarrow \mathsf{aProve}(x, w, \mathsf{dk}, \mathsf{am})\end{array} : \begin{array}{l}\mathsf{Verify}(x, \pi^*) = 1 \\ \mathsf{aExtract}(x, \pi^*, \mathsf{dk}) = \mathsf{am}\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

3.2 Security notions of ANIZK

We now formalize the security notions for ANIZK, namely, anamorphic security, weak robustness, strong robustness, recipient anamorphic zero knowledge, recipient anamorphic simulation soundness, and recipient anamorphic simulation extractability.

Since the first three algorithms of ANIZK (i.e., $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$) form a standard NIZK, these algorithms are required to satisfy ordinary completeness, computational soundness, and zero knowledge as defined in Definition 4.

Anamorphic Security. It requires the anamorphic channel to remain hidden even when a prover is asked to reveal the randomness associated with each proof. An efficient equivocation algorithm supplies alternative randomness that explains an anamorphic proof as a legitimate output of the ordinary proving algorithm. No PPT adversary without the double key can distinguish an anamorphic proof paired with this alternative randomness from an ordinary proof accompanied by its true randomness.

Definition 11 (Anamorphic Security). *An ANIZK scheme* ANIZK *is anamorphically secure, if there exists a PPT algorithm* Equiv *(which takes* $(x, w, \mathsf{dk}, \mathsf{am},$

[7] We also assume that these algorithms implicitly include a common reference string crs as an input.

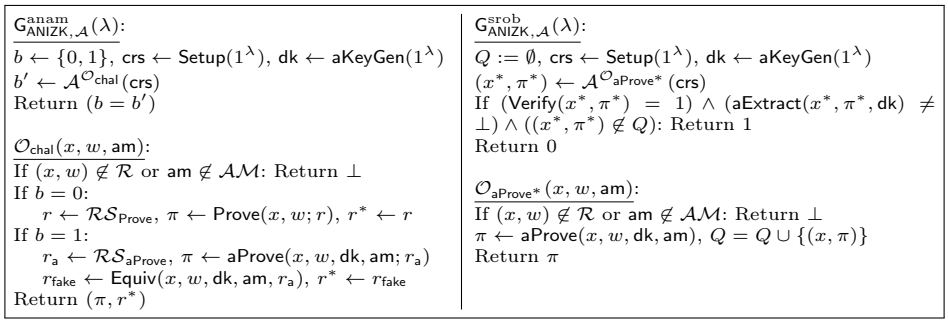

Fig. 4: Anamorphic security, strong robustness games for ANIZK

$r_{\mathsf{a}} \in \mathcal{RS}_{\mathsf{aProve}}$) *as input and outputs* $r_{\mathsf{fake}} \in \mathcal{RS}_{\mathsf{Prove}}$) *such that for any PPT adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\mathrm{anam}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda) = |\Pr[\mathsf{G}^{\mathrm{anam}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\mathrm{anam}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda)$ *is defined in Fig. 4.*

Strong Robustness. The notion of strong robustness requires that an adversary without the double key cannot produce a fresh valid proof from which a non-empty anamorphic message can be extracted, even after obtaining adaptively chosen anamorphic proofs. Analogous notions have been explored in the context of anamorphic encryption [7] and signatures [11].

Definition 12 (Strong Robustness). *An ANIZK scheme* ANIZK *is strongly robust, if for any PPT adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\mathrm{srob}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda) = \Pr[\mathsf{G}^{\mathrm{srob}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\mathrm{srob}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda)$ *is defined in Fig. 4.*

We also consider the notion of weak robustness for ANIZK. Compared to strong robustness, it restricts the challenge proof π^* to be honestly generated using Prove. Since strong robustness implies weak robustness, we provide the formal definition of weak robustness in Appendix B.

Recipient Anamorphic Zero Knowledge. This property ensures that, upon receiving an anamorphic proof, a recipient holding the double key learns nothing about the witness beyond the validity of the statement and the embedded anamorphic message.

Definition 13 (Recipient Anamorphic Zero Knowledge). *An ANIZK scheme* ANIZK *is recipient anamorphic zero-knowledge, if there exists a PPT simulator* $\mathcal{S}_{\mathsf{ana}} = (\mathcal{S}_{\mathsf{ana}}.\mathsf{Setup}, \mathcal{S}_{\mathsf{ana}}.\mathsf{aProve})$ *such that for any PPT adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\mathrm{razk}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda) = |\Pr[\mathsf{G}^{\text{razk-real}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{G}^{\text{razk-sim}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\text{razk-real}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda)$ *and* $\mathsf{G}^{\text{razk-sim}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda)$ *are defined in Fig. 5.*

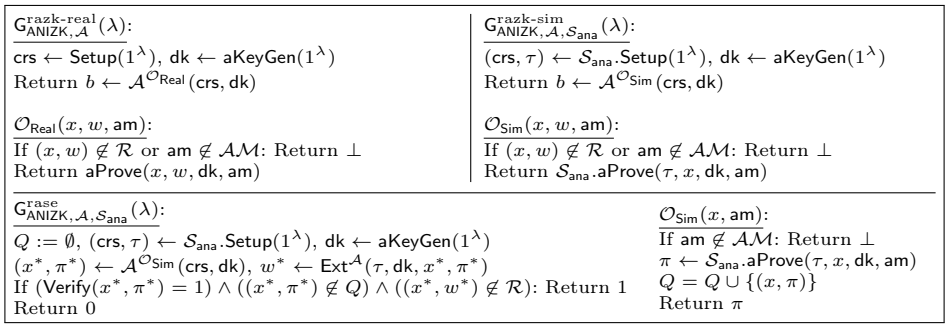

Fig. 5: Recipient anamorphic zero knowledge and recipient anamorphic simulation extractability game for ANIZK

Recipient Anamorphic Simulation Extractability. This property requires that, even when given the double key and adaptive access to simulated anamorphic proofs, an adversary cannot output a fresh accepting statement-proof pair unless a valid witness for that statement can be efficiently extracted.

Definition 14 (Recipient Anamorphic Simulation Extractability). *An ANIZK scheme* ANIZK*, with a PPT simulator* $\mathcal{S}_{\mathsf{ana}} = (\mathcal{S}_{\mathsf{ana}}.\mathsf{Setup}, \mathcal{S}_{\mathsf{ana}}.\mathsf{aProve})$ *that satisfies Definition 13, is recipient anamorphic simulation extractable, if for any PPT adversary* $\mathcal{A}$*, there exists a PPT extractor* Ext*, such that*

$$\mathbf{Adv}^{\text{rase}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda) = \Pr[\mathsf{G}^{\text{rase}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\text{rase}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda)$ *is defined in Fig. 5.*

We also consider recipient anamorphic simulation soundness for ANIZK, where the adversary simply wins by outputting a forgery $(x^*, \pi^*) \notin Q$ such that $x^* \notin \mathcal{L}_{\mathcal{R}}$ and $\mathsf{Verify}(x^*, \pi^*) = 1$. Since it is naturally implied by recipient anamorphic simulation extractability, we defer the formal definition to Appendix B.

4 Anamorphic NIZK Construction

In this section, we present a construction of ANIZK and show that it achieves the required security properties.

Let $\mathsf{Sigma} = (\mathsf{P}, \mathsf{V})$ be a Sigma protocol, with challenge space $\mathcal{CL}$ and commitment space $\mathcal{CM}$, for a polynomial-time relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$. Let $\mathsf{AEAD} = (\mathsf{AEAD.Enc}, \mathsf{AEAD.Dec})$ be an AEAD scheme with message space $\mathcal{M}$, key space $\mathcal{K} = \{0,1\}^\lambda$, associated data space $\mathcal{AD} = \mathcal{X} \times \mathcal{CM}$, and ciphertext space $\mathcal{CT} = \{0,1\}^{\ell(\lambda)}$ for some polynomial $\ell(\cdot)$. Furthermore, let $\mathcal{H}$ be a hash function family where each $\mathsf{H} \in \mathcal{H}$ is defined as $\mathsf{H} : \mathcal{X} \times \mathcal{CM} \times \{0,1\}^{\ell(\lambda)} \to \mathcal{CL}$.

Our generic ANIZK scheme $\mathsf{ANIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{aKeyGen}, \mathsf{aProve}, \mathsf{aExtract})$, with anamorphic message space $\mathcal{AM} = \mathcal{M}$, for relation $\mathcal{R}$ is described in Fig. 6.

$\underline{\mathsf{Setup}(1^\lambda)}$:
$\mathsf{H} \leftarrow \mathcal{H}$
Return $\mathsf{crs} := \mathsf{H}$

$\underline{\mathsf{Prove}(x, w)}$:
$(a, \mathsf{st}) \leftarrow \mathsf{P}_1(x, w)$, $S \leftarrow \{0,1\}^{\ell(\lambda)}$
$c = \mathsf{H}(x, a, S)$, $z \leftarrow \mathsf{P}_2(a, c, x, w, \mathsf{st})$
Return $\pi := (a, S, z)$

$\underline{\mathsf{Verify}(x, \pi)}$:
Parse $\pi = (a, S, z)$
$c \leftarrow \mathsf{H}(x, a, S)$
If $\mathsf{V}_2(x, a, c, z) = 1$: Return 1; Else: Return 0

$\underline{\mathsf{aKeyGen}(1^\lambda)}$:
Return $\mathsf{dk} \leftarrow \{0,1\}^\lambda$

$\underline{\mathsf{aProve}(x, w, \mathsf{dk}, \mathsf{am})}$:
$(a, \mathsf{st}) \leftarrow \mathsf{P}_1(x, w)$
$S \leftarrow \mathsf{AEAD.Enc}(\mathsf{dk}, \mathsf{am}, \mathsf{ad} = (x, a))$
$c = \mathsf{H}(x, a, S)$, $z \leftarrow \mathsf{P}_2(a, c, x, w, \mathsf{st})$
Return $\pi := (a, S, z)$

$\underline{\mathsf{aExtract}(x, \pi, \mathsf{dk})}$:
Parse $\pi = (a, S, z)$
Return $\mathsf{AEAD.Dec}(\mathsf{dk}, S, \mathsf{ad} = (x, a))$

Fig. 6: The generic construction of ANIZK

The correctness of ANIZK is trivially guaranteed by the correctness of AEAD and Sigma.

We establish its security in the following theorem.

Theorem 1. *If the Sigma protocol* (P, V) *satisfies completeness, special soundness, special HVZK, quasi-unique responses, and commitment unpredictability, and has a challenge space* $\mathcal{CL}$ *of super-polynomial size (i.e,* $|\mathcal{CL}| = \lambda^{\omega(1)}$*), and the AEAD scheme* AEAD *provides IND$-CPA and INT-CTXT security, then* ANIZK *in Fig. 6 achieves all security properties in Section 3.2.*

Proof. We show the proof of recipient anamorphic zero knowledge and recipient anamorphic simulation extractability here, and defer the rest to Appendix C due to space limitations.

Proof of Recipient Anamorphic Zero Knowledge. We now prove the recipient anamorphic zero knowledge (RAZK) property in the random oracle model, where the hash function H is modeled as a random oracle. For any PPT adversary $\mathcal{A}$, let q_H and q_P denote the upper bounds on the number of its direct random oracle queries to H and valid proof queries to the real/simulated proof oracles, respectively.

To ensure the freshness of random oracle programming, we rely on the commitment unpredictability of the underlying Sigma protocol. Specifically, the probability of generating any specific commitment a is bounded by $2^{-\mu(\lambda)}$ (where $\mu(\lambda) = \omega(\log \lambda)$). By the special-HVZK property, the simulated commitment a output by $(a, z) \leftarrow \mathcal{S}_{\mathsf{Sigma}}(x, c)$ achieves the same min-entropy bound.

Let $\mathcal{S}_{\mathsf{Sigma}}$ be the special-HVZK simulator of the Sigma protocol. We construct the RAZK simulator $\mathcal{S}_{\mathsf{ana}} = (\mathcal{S}_{\mathsf{ana}}.\mathsf{Setup}, \mathcal{S}_{\mathsf{ana}}.\mathsf{aProve})$ as follows:

- $\underline{\mathcal{S}_{\mathsf{ana}}.\mathsf{Setup}(1^\lambda)}$: It initializes an empty random oracle table $L_{\mathsf{H}} := \emptyset$. It outputs the public parameter crs granting access to this table, and stores L_{H} along with the programming capability in the simulation state τ.
- $\underline{\mathcal{S}_{\mathsf{ana}}.\mathsf{aProve}(\tau, x, \mathsf{dk}, \mathsf{am})}$: It samples $c \leftarrow \mathcal{CL}$, runs $(a, z) \leftarrow \mathcal{S}_{\mathsf{Sigma}}(x, c)$, and computes the ciphertext $S \leftarrow \mathsf{AEAD.Enc}(\mathsf{dk}, \mathsf{am}, \mathsf{ad} = (x, a))$. If $L_{\mathsf{H}}[x, a, S]$ is undefined, it programs $L_{\mathsf{H}}[x, a, S] := c$ and returns $\pi = (a, S, z)$. If $L_{\mathsf{H}}[x, a, S]$ is already defined, it aborts and outputs $\perp$.

We proceed via a sequence of games. Let E_i denote the event that $\mathcal{A}$ outputs 1 in Game i.

Game 0: This is the real game $\mathsf{G}^{\text{razk-real}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda)$. The challenger initializes $L_{\mathsf{H}} := \emptyset$, generates $\mathsf{dk} \leftarrow \mathsf{aKeyGen}(1^\lambda)$, and gives $(\mathsf{crs}, \mathsf{dk})$ to $\mathcal{A}$. The random oracle H is simulated via lazy sampling. On a valid query $\mathcal{O}_{\mathsf{Real}}(x, w, \mathsf{am})$, the challenger generates an honest proof $\pi = (a, S, z)$ by running P_1, evaluating $\mathsf{H}(x, a, S)$ directly to get c, running P_2, and returning (a, S, z). By definition, $\Pr[E_0] = \Pr[\mathsf{G}^{\text{razk-real}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda) = 1]$.

Game 1: This game modifies Game 0 by explicitly programming the random oracle during proof generation. For each valid query to $\mathcal{O}_{\mathsf{Real}}$, after generating $(a, \mathsf{st}) \leftarrow \mathsf{P}_1(x, w)$ and $S \leftarrow \mathsf{AEAD.Enc}(\mathsf{dk}, \mathsf{am}, \mathsf{ad} = (x, a))$, the challenger checks if $L_{\mathsf{H}}[x, a, S]$ is already defined. If so, it triggers an abort event Bad_1. Otherwise, it samples $c \leftarrow \mathcal{CL}$, explicitly programs $L_{\mathsf{H}}[x, a, S] := c$, computes $z \leftarrow \mathsf{P}_2(a, c, x, w, \mathsf{st})$, and returns (a, S, z).

Since lazy sampling and explicit programming produce identical distributions when the programmed point is fresh, from $\mathcal{A}$'s point of view, Game 0 and Game 1 are identical unless Bad_1 occurs. Prior to generating the j-th proof, the table L_{H} contains at most $q_H + j - 1$ entries. Even though $\mathcal{A}$ knows dk and am, the commitment a generated by P_1 has conditional min-entropy $\mu(\lambda)$. The probability that the newly generated (x, a, S) collides with any existing entry (x, a', S') in L_{H} is bounded by the probability that $a = a'$, which is at most $2^{-\mu(\lambda)}$. Taking a union bound over the table size and all q_P proof queries, we have

$$\left|\Pr[E_0] - \Pr[E_1]\right| \leq \Pr[\mathsf{Bad}_1] \leq q_P(q_H + q_P)2^{-\mu(\lambda)} \leq \mathsf{negl}(\lambda). \tag{1}$$

Game 2: This game modifies Game 1 by replacing the honest Sigma prover with the SHVZK simulator. For each valid query, the challenger samples $c \leftarrow \mathcal{CL}$, runs $(a, z) \leftarrow \mathcal{S}_{\mathsf{Sigma}}(x, c)$, and computes $S \leftarrow \mathsf{AEAD.Enc}(\mathsf{dk}, \mathsf{am}, \mathsf{ad} = (x, a))$. It then performs the exact same abort check on $L_{\mathsf{H}}[x, a, S]$, programs it to c, and returns (a, S, z). Notice that the responses in Game 2 perfectly match the output of the simulator $\mathcal{S}_{\mathsf{ana}}.\mathsf{aProve}$, hence Game 2 is precisely the simulated game $\mathsf{G}^{\text{razk-sim}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda)$.

We bridge Game 1 and Game 2 via intermediate hybrids $\mathsf{Hyb}_0, \ldots, \mathsf{Hyb}_{q_P}$, where Hyb_0 is Game 1 and Hyb_{q_P} is Game 2. In Hyb_j, the first j valid proof queries use simulated Sigma transcripts, and the remaining queries use honest transcripts. For each $j \in \{1, \ldots, q_P\}$, we can construct a special-HVZK adversary $\mathcal{D}_j$ that distinguishes Hyb_{j-1} from Hyb_j. $\mathcal{D}_j$ gives dk to $\mathcal{A}$, simulates all random oracle queries via L_{H}, and answers all proof queries itself except for the j-th valid query, which it forwards to its SHVZK challenger. Upon receiving the challenge transcript (a^*, c^*, z^*), $\mathcal{D}_j$ computes S^*, applies the identical abort check on L_{H}, programs it, and returns the proof to $\mathcal{A}$. Since the abort condition strictly relies on a and S, and $\mathcal{D}_j$ perfectly simulates the surrounding environment, a standard reduction yields

$$\left|\Pr[E_1] - \Pr[E_2]\right| \leq \sum_{j=1}^{q_P} \mathbf{Adv}^{\text{shvzk}}_{\mathsf{Sigma},\mathcal{D}_j,\mathcal{S}_{\mathsf{Sigma}}}(\lambda). \tag{2}$$

Combining the bounds from equations (1) and (2), we obtain

$$\mathbf{Adv}^{\mathrm{razk}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda) = \big|\Pr[E_0] - \Pr[E_2]\big| \\ \leq q_P(q_H + q_P)2^{-\mu(\lambda)} + \sum_{j=1}^{q_P} \mathbf{Adv}^{\mathrm{shvzk}}_{\mathsf{Sigma},\mathcal{D}_j,\mathcal{S}_{\mathsf{Sigma}}}(\lambda) \leq \mathsf{negl}(\lambda).$$

This concludes the proof. □

Proof of Recipient Anamorphic Simulation Extractability. We prove the recipient anamorphic simulation extractability (RASE) property in the random oracle model (ROM). We use the RAZK simulator $\mathcal{S}_{\mathsf{ana}} = (\mathcal{S}_{\mathsf{ana}}.\mathsf{Setup}, \mathcal{S}_{\mathsf{ana}}.\mathsf{aProve})$ constructed previously. For any PPT adversary $\mathcal{A}$, let q_H and q_S denote the upper bounds on the number of its direct random oracle queries to H and valid simulation queries to $\mathcal{O}_{\mathsf{Sim}}$, respectively.

Extractor Construction. We now turn to the construction of the extractor $\mathsf{Ext}^{\mathcal{A}}$.

Note that since the extractor $\mathsf{Ext}^{\mathcal{A}}$ is granted black-box access to the adversary $\mathcal{A}$, it inherently acts as the execution environment for $\mathcal{A}$. Consequently, $\mathsf{Ext}^{\mathcal{A}}$ controls $\mathcal{A}$'s random coins and internally maintains the complete execution transcript, including the state of the random oracle table L_{H} and the history of all queries made to H and $\mathcal{O}_{\mathsf{Sim}}$. The simulation state τ merely grants the initial programming capability for L_{H}. On input $(\tau, \mathsf{dk}, x^*, \pi^* = (a^*, S^*, z^*))$, the extractor $\mathsf{Ext}^{\mathcal{A}}$ proceeds as follows:

1. It first checks whether $\mathsf{V}_2(x^*, a^*, c^*, z^*) = 1$, where $c^* = L_{\mathsf{H}}[x^*, a^*, S^*]$. If not, it outputs $\perp$.
2. It checks its recorded query transcript to identify the origin of the assignment for c^*. If c^* was programmed by the simulation oracle $\mathcal{O}_{\mathsf{Sim}}$ rather than generated via a direct H query (or implicit lazy sampling), it outputs $\perp$.
3. Otherwise, it applies the General Forking Lemma (Lemma 1) by rewinding $\mathcal{A}$ to the exact point where (x^*, a^*, S^*) was queried to H. It replays $\mathcal{A}$ with fresh independent random oracle responses from that point onward.
4. If a second accepting proof $\pi' = (a^*, S^*, z')$ for x^* is obtained with a distinct challenge $c' \neq c^*$, the extractor runs the special soundness extractor of the underlying Sigma to compute and output $w^* \leftarrow \mathsf{Ext}_{\mathsf{Sigma}}(x^*, a^*, c^*, z^*, c', z')$. To ensure the extractor runs in strict polynomial time, this rewinding process is repeated at most $N(\lambda)$ times (where $N(\lambda)$ is a specific polynomial bounded below), outputting $\perp$ if all attempts fail.

By the special soundness property, whenever $\mathsf{Ext}_{\mathsf{Sigma}}$ successfully outputs a witness w^*, it is unconditionally guaranteed that $(x^*, w^*) \in \mathcal{R}$.

Analysis of $\mathcal{A}$'s Advantage. Let Q be the set of all simulated proof pairs $(x_i, \pi_i = (a_i, S_i, z_i))$ returned by $\mathcal{O}_{\mathsf{Sim}}$. For convenience, we define

$$Q_{\mathrm{prx}} = \{(x, a, S) : \exists z \text{ such that } (x, (a, S, z)) \in Q\}.$$

Let $\epsilon = \mathbf{Adv}^{\mathrm{rase}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda)$ be the probability that $\mathcal{A}$ wins the RASE game. The winning condition requires $\mathcal{A}$ to output a valid proof (x^*, π^*) such that $(x^*, \pi^*) \notin Q$ and $(x^*, w^*) \notin \mathcal{R}$.

Recall that the simulator $\mathcal{S}_{\mathsf{ana}}$ aborts if a freshly generated tuple (x, a, S) collides with an existing entry in L_{H}. Based on the min-entropy bound $\mu(\lambda)$ of the underlying Sigma protocol, the probability of such a simulation failure across all queries and rewinds is bounded by $\delta_{\mathsf{prog}} = N(\lambda)q_S(q_H + q_S)2^{-\mu(\lambda)} = \mathsf{negl}(\lambda)$. Assuming no simulation abort occurs, we partition $\mathcal{A}$'s successful forgery into two mutually exclusive types:

Type 1: Prefix Reuse. The adversary wins and $(x^*, a^*, S^*) \in Q_{\mathrm{prx}}$. Since $(x^*, a^*, S^*) \in Q_{\mathrm{prx}}$, there exists a simulated proof $(x^*, (a^*, S^*, z_i)) \in Q$. Both π^* and the simulated proof share the same challenge $c^* = L_{\mathsf{H}}[x^*, a^*, S^*]$. However, the winning condition requires $(x^*, \pi^*) \notin Q$, which implies $z^* \neq z_i$. This means $\mathcal{A}$ has found two distinct accepting responses z^* and z_i for the exact same commitment and challenge (x^*, a^*, c^*). We can straightforwardly construct an adversary $\mathcal{B}$ that outputs $(x^*, a^*, c^*, z^*, z_i)$ to break the quasi-unique responses property of the Sigma protocol. Thus, the probability of a Type 1 forgery is bounded by $\mathbf{Adv}^{\text{qur-uni}}_{\mathsf{Sigma},\mathcal{B}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Type 2: Fresh Prefix. The adversary wins and $(x^*, a^*, S^*) \notin Q_{\mathrm{prx}}$. Suppose towards a contradiction that there exists a PPT adversary $\mathcal{A}$ whose probability of a Type 2 forgery, $\varepsilon_{\mathsf{fresh}}$, is non-negligible. Since $\varepsilon_{\mathsf{fresh}}$ is non-negligible, there exists a polynomial $p(\lambda)$ such that $\varepsilon_{\mathsf{fresh}} \geq 1/p(\lambda)$ for infinitely many λ.

Since the prefix was never programmed by $\mathcal{O}_{\mathsf{Sim}}$, the challenge c^* must have been generated by a standard random oracle query. The total number of relevant random oracle evaluations is bounded by $q \leq q_H + q_S + 1$, which is strictly polynomial. By the General Forking Lemma, a single execution of the rewinding procedure yields two accepting transcripts with distinct challenges $c^* \neq c'$ with probability at least:

$$\varepsilon_{\mathsf{fork}} \geq \varepsilon_{\mathsf{fresh}} \left(\frac{\varepsilon_{\mathsf{fresh}}}{q} - \frac{1}{|\mathcal{CL}|} \right).$$

Since $\varepsilon_{\mathsf{fresh}} \geq 1/p(\lambda)$ and $|\mathcal{CL}|$ is super-polynomial, the single-attempt success probability $\varepsilon_{\mathsf{fork}} \geq 1/p'(\lambda)$ for some polynomial p'.

Since the RASE definition specifies that the extractor $\mathsf{Ext}^{\mathcal{A}}$ is granted black-box access to the specific adversary $\mathcal{A}$, we can define $\mathsf{Ext}^{\mathcal{A}}$ to repeat the forking procedure $N(\lambda) = \lambda \cdot p'(\lambda)$ times. This runs in strict polynomial time. The probability that all $N(\lambda)$ independent attempts fail is:

$$\delta_{\mathsf{fork}}(\lambda) = (1 - \varepsilon_{\mathsf{fork}})^{N(\lambda)} \leq \left(1 - \frac{1}{p'(\lambda)}\right)^{\lambda \cdot p'(\lambda)} \leq e^{-\lambda} = \mathsf{negl}(\lambda).$$

Therefore, the extractor successfully applies $\mathsf{Ext}_{\mathsf{Sigma}}$ to output a witness w^* except with negligible probability $\delta_{\mathsf{fork}}(\lambda)$.

As established earlier, whenever extraction succeeds, special soundness strictly guarantees that $(x^*, w^*) \in \mathcal{R}$. However, the winning condition of the RASE game explicitly requires $(x^*, w^*) \notin \mathcal{R}$. This creates a direct contradiction. Consequently, the adversary can only win if the extractor simultaneously fails to fork. Therefore, the adversary's winning probability is upper-bounded by the

extractor's failure probability:

$$\varepsilon_{\mathsf{fresh}} \leq \Pr[\text{Extractor Fails}] = \delta_{\mathsf{fork}}(\lambda) = \mathsf{negl}(\lambda).$$

This directly contradicts the initial assumption that $\varepsilon_{\mathsf{fresh}}$ is non-negligible.

Combining the bounds from both types and the simulation failure probability, we conclude that the overall advantage of the adversary is:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{rase}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda) &\leq \delta_{\mathsf{prog}} + \mathbf{Adv}^{\text{qur-uni}}_{\mathsf{Sigma},\mathcal{B}}(\lambda) + \delta_{\mathsf{fork}}(\lambda) \\ &\leq \mathsf{negl}(\lambda).\end{aligned}$$

This concludes the proof that the proposed scheme satisfies recipient anamorphic simulation extractability. □

In addition to completeness, special soundness, and special HVZK, our construction requires the underlying Sigma protocol to satisfy quasi-unique response property and commitment unpredictability. These additional properties are satisfied by many standard Sigma protocols. Moreover, commitment unpredictability can be ensured by appending an independent, uniformly random λ-bit string to the first message, while preserving the other required properties [9]. Our construction therefore applies to a broad class of standard Sigma protocols, including those that satisfy the remaining assumptions and require only this simple modification.

5 Application: Recipient Anamorphic Encryption

In this section, we explore an application of ANIZK to recipient anamorphic encryption (RAME) in the symmetric setting. We first review the basic syntax of RAME. Next, we present a generic RAME construction from ANIZK and establish its security.

RAME. We recall the definition of RAME [16,7] as follows.

Definition 15 (RAME). *An RAME scheme* $\mathsf{RAME} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{aKeyGen}, \mathsf{aEnc}, \mathsf{aDec})$ *consists of a public-key encryption scheme* $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *and an associated anamorphic triplet* $(\mathsf{aKeyGen}, \mathsf{aEnc}, \mathsf{aDec})$. RAME *operates in two modes: the normal mode and the anamorphic mode. In the normal mode, the primitive is identical to* PKE*; in the anamorphic mode,* PKE *is replaced by* $(\mathsf{aKeyGen}, \mathsf{aEnc}, \mathsf{aDec})$*, where the algorithms are defined as follows:*

- $\mathsf{aKeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk}, \mathsf{dk})$*: The anamorphic key generation algorithm takes the security parameter* λ *as input, and then outputs a public key* pk*, a secret key* sk *and a double key* dk*.*
- $\mathsf{aEnc}(\mathsf{pk}, \mathsf{dk}, \mathsf{m}, \mathsf{am}) \to \mathsf{act}$*: The anamorphic encryption algorithm takes a public key* pk*, a double key* dk*, a normal message* m *and an anamorphic message* am *as input, and then outputs an anamorphic ciphertext* act*.*
- $\mathsf{aDec}(\mathsf{pk}, \mathsf{dk}, \mathsf{act}) \to \mathsf{am}/\perp$*: The anamorphic decryption algorithm takes a public key* pk*, a double key* dk *and an anamorphic ciphertext* act*, and then outputs an anamorphic message* am *or a fail symbol* $\perp$*.*

Correctness. For every (all-powerful) adversary $\mathcal{A}$, the probability

$$\Pr\left[\begin{array}{l|l} (\mathsf{pk},\mathsf{sk},\mathsf{dk}) \leftarrow \mathsf{aKeyGen}(1^\lambda) & \mathsf{aDec}(\mathsf{pk},\mathsf{dk},\mathsf{act}) = \mathsf{am} \\ (\mathsf{m},\mathsf{am}) \leftarrow \mathcal{A}(\mathsf{pk}) & : \mathsf{Dec}(\mathsf{sk},\mathsf{ct}) = \mathsf{m} \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk},\mathsf{m}) & \mathsf{Dec}(\mathsf{sk},\mathsf{act}) = \mathsf{m} \\ \mathsf{act} \leftarrow \mathsf{aEnc}(\mathsf{pk},\mathsf{dk},\mathsf{m},\mathsf{am}) & \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

To formally capture these requirements in the presence of decryption queries, we adopt the security notions introduced by [7]. Specifically, STE-xCCA1 ensures the computational indistinguishability between normal and anamorphic ciphertexts against an adversary equipped with an anamorphic decryption oracle. Complementarily, SIND-aCCA1 and SIND-nCCA guarantee the confidentiality of the underlying ordinary messages within anamorphic and normal ciphertexts, respectively, even when the adversary is granted standard decryption queries. As shown in [7, Theorem 16], satisfying STE-xCCA1 implies both anamorphic security and strong robustness.

Construction of RAME from ANIZK. We now present a generic construction of RAME and show that it achieves the required security properties.

Let $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme with message space $\mathcal{M}_{\mathsf{PKE}}$, as defined in Definition 19. Let $\mathsf{ANIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{aKeyGen}, \mathsf{aProve}, \mathsf{aExtract})$ be an ANIZK scheme with anamorphic message space $\mathcal{AM}_{\mathsf{ANIZK}}$ for the following relation:

$$\mathcal{R}_{\mathsf{NY}} = \left\{ \begin{pmatrix} x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2), \\ w = (\mathsf{m}, r_1, r_2) \end{pmatrix} : \begin{array}{l} \mathsf{ct}_1 = \mathsf{PKE.Enc}(\mathsf{pk}_1, \mathsf{m}; r_1) \\ \wedge\ \mathsf{ct}_2 = \mathsf{PKE.Enc}(\mathsf{pk}_2, \mathsf{m}; r_2) \end{array} \right\}. \tag{3}$$

Our RAME scheme $\mathsf{RAME} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{aKeyGen}, \mathsf{aEnc}, \mathsf{aDec})$, with message space $\mathcal{M} = \mathcal{M}_{\mathsf{PKE}}$ and anamorphic message space $\mathcal{AM} = \mathcal{AM}_{\mathsf{ANIZK}}$, is described in Fig. 7.

The correctness of RAME is straightforward. We establish its security in the following theorem, the proof of which is given in Appendix E due to space constraints.

Theorem 2. *If* PKE *satisfies perfect correctness and IND-CPA security, and* ANIZK *for* $\mathcal{R}_{\mathsf{NY}}$ *satisfies all security properties in Section 3.2, then* RAME *in Fig. 7 achieves STE-xCCA1 and SIND-aCCA1 security. Furthermore, if the NIZK formed by the first three algorithms of* ANIZK *(i.e.,* ANIZK.Setup, ANIZK.Prove, *and* ANIZK.Verify*) satisfies simulation soundness (Definition 18), then* RAME *also achieves SIND-nCCA security.*

ANIZK has potential applications in a broad range of cryptographic settings, including anamorphic encryption, anamorphic signatures, and anamorphic key exchange. In this work, we focus on RAME as a concrete example illustrating how ANIZK can serve as a building block for cryptographic protocols with covert communication capabilities. This construction demonstrates one application of ANIZK, while its use in other settings remains a promising direction for future work.

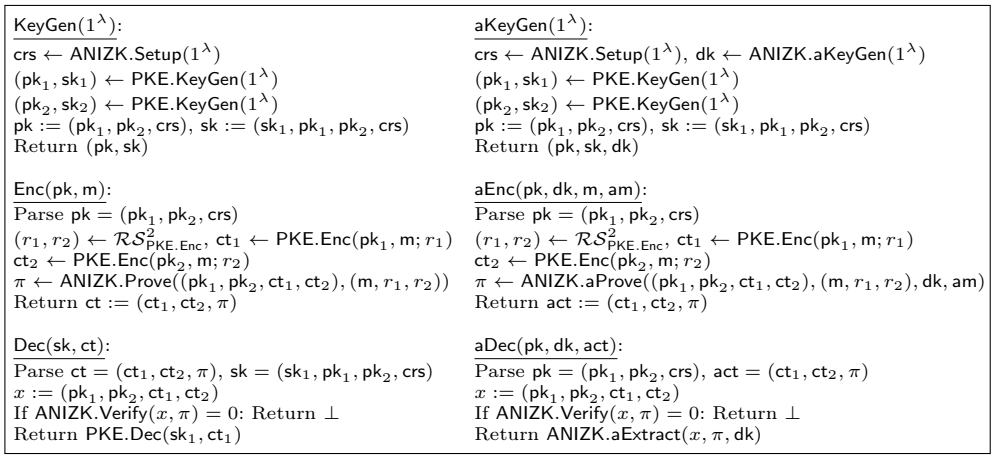

Fig. 7: Construction of RAME based on ANIZK

6 Asymmetric Anamorphic NIZK

In this section, we introduce an asymmetric anamorphic non-interactive zero-knowledge proof system (ASY-ANIZK). Unlike ANIZK, which uses a shared double key dk for both message embedding and extraction, ASY-ANIZK separates these capabilities through an anamorphic key pair (apk, ask). A prover holding a valid witness can use the public key apk to embed an anamorphic message into a proof, whereas only the recipient holding the secret key ask can recover it. This asymmetric design enables covert communication to a designated recipient without requiring the prover and recipient to share a secret key. We formally define ASY-ANIZK below.

Definition 16 (Asymmetric Anamorphic NIZK). *An ASY-ANIZK scheme* ASY-ANIZK = (Setup, Prove, Verify, aKeyGen, aProve, aExtract) *consists of a standard NIZK proof system* NIZK = (Setup, Prove, Verify)*, along with three additional algorithms* (aKeyGen, aProve, aExtract) *defined as follows:*

- aKeyGen(1^λ) $\to$ (apk, ask) : *The anamorphic key generation algorithm takes the security parameter λ as input, and then outputs an anamorphic public key* apk *and an anamorphic secret key* ask*.*
- aProve(x, w, apk, am) $\to \pi^*$: *The anamorphic proving algorithm takes a statement x, a witness w, an anamorphic public key* apk*, and an anamorphic message* am *as input, where $(x, w) \in \mathcal{R}$, and then outputs a proof π^*.*
- aExtract(x, π^*, ask) $\to$ am/$\perp$: *The anamorphic extraction algorithm takes a statement x, a proof π^* and an anamorphic secret key* ask *as input, and then outputs an anamorphic message* am *or a fail symbol $\perp$.*

and satisfies the following properties:

$\underline{\mathsf{G}^{\mathrm{wrob}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda)}$:
$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{aKeyGen}(1^\lambda)$, $(x, w) \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{apk})$
If $(x, w) \notin \mathcal{R}$: Return $\perp$
$\pi \leftarrow \mathsf{Prove}(x, w)$, $\mathsf{am} \leftarrow \mathsf{aExtract}(x, \pi, \mathsf{ask})$
Return 1 if and only if $\mathsf{am} \neq \perp$

Fig. 8: Recipient anamorphic weak robustness game for ASY-ANIZK

Anamorphic correctness. For all $\mathsf{am} \in \mathcal{AM}$*, all* $(x, w) \in \mathcal{R}$*,*

$$\Pr\left[\begin{array}{l|l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) & \\ (\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{aKeyGen}(1^\lambda) & : \begin{array}{l} \mathsf{Verify}(x, \pi^*) = 1 \\ \mathsf{aExtract}(x, \pi^*, \mathsf{ask}) = \mathsf{am} \end{array} \\ \pi^* \leftarrow \mathsf{aProve}(x, w, \mathsf{apk}, \mathsf{am}) & \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Unlike ANIZK, ASY-ANIZK does not satisfy strong robustness. Since the anamorphic public key apk is publicly available, an adversary can use apk to embed any nonempty anamorphic message into a freshly generated proof. Such a proof is accepted by the verification algorithm, and the embedded message can subsequently be recovered by the recipient using the anamorphic extraction algorithm $\mathsf{aExtract}$. This violates the strong robustness requirement. Therefore, ASY-ANIZK considers weak robustness instead.

The other three notions, namely anamorphic security, recipient anamorphic zero-knowledge, and recipient anamorphic simulation extractability, are defined analogously to their ANIZK counterparts. Due to space limitations, we defer their formal definitions to Appendix F.1.

Now we show the weak robustness as follows. This property ensures that an honestly generated ordinary proof is not mistakenly interpreted as carrying an anamorphic message when processed using an independently generated anamorphic secret key.

Definition 17 (Weak Robustness). *An ASY-ANIZK scheme* $\mathsf{ASY\text{-}ANIZK}$ *is weakly robust, if for any PPT adversary,*

$$\mathbf{Adv}^{\mathrm{wrob}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda) = \Pr[\mathsf{G}^{\mathrm{wrob}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\mathrm{wrob}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda)$ *is defined in Fig. 8.*

Construction of ASY-ANIZK. We present an ASY-ANIZK scheme as follows.

Let $\mathsf{LNIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ be a labeled NIZK scheme as defined in Definition 31, with label space $\mathcal{LS}$, for a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$. Let $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme as defined in Definition 19, with message space $\mathcal{M}$ and ciphertext space $\mathcal{CT} = \mathcal{LS}$.

Our ASY-ANIZK scheme $\mathsf{ASY\text{-}ANIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{aKeyGen}, \mathsf{aProve}, \mathsf{aExtract})$, with anamorphic message space $\mathcal{AM} = \mathcal{M}$, for relation $\mathcal{R}$ is shown in Fig. 9.

The anamorphic correctness of $\mathsf{ASY\text{-}ANIZK}$ follows immediately from the perfect completeness of LNIZK and the decryption correctness of PKE. We establish

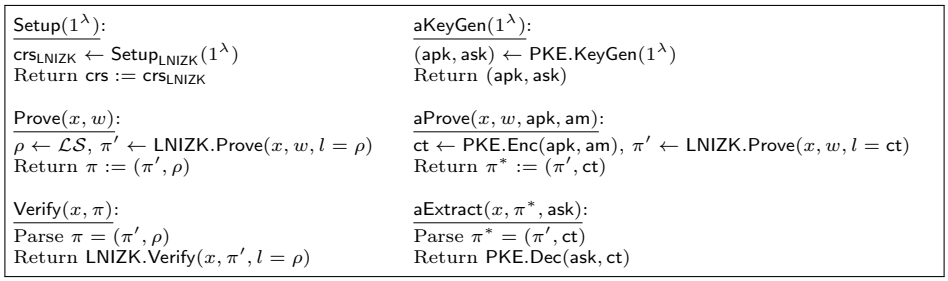

Fig. 9: The generic construction of ASY-ANIZK

its security in the following theorem, deferring the formal proof to Appendix H due to space limitations.

Theorem 3. *If the PKE scheme* PKE *is IND$-CPA secure and satisfies random ciphertext rejection, and the LNIZK scheme* LNIZK *satisfies completeness, labeled zero-knowledge, and simulation extractability, then* ASY-ANIZK *in Fig. 9 satisfies anamorphic security, weak robustness, recipient anamorphic zero knowledge, and recipient anamorphic simulation extractability.*

References

1. Banerjee, S., Pal, T., Rupp, A., Slamanig, D.: Simple asymmetric anamorphic encryption and signature using multi-message extensions. In: Advances in Cryptology – CRYPTO 2026. Springer (2026), https://eprint.iacr.org/2025/370, full version available as Cryptology ePrint Archive, Paper 2025/370.
2. Banfi, F., Gegier, K., Hirt, M., Maurer, U., Rito, G.: Anamorphic encryption, revisited. In: Advances in Cryptology – EUROCRYPT 2024. Lecture Notes in Computer Science, vol. 14652, pp. 3–32. Springer (2024). https://doi.org/10.1007/978-3-031-58723-8_1, https://doi.org/10.1007/978-3-031-58723-8_1
3. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: CCS 2006. pp. 390–399 (2006)
4. Beullens, W., Dobson, S., Katsumata, S., Lai, Y., Pintore, F.: Group signatures and more from isogenies and lattices: Generic, simple, and efficient. In: Advances in Cryptology - EUROCRYPT 2022. Lecture Notes in Computer Science, vol. 13276, pp. 95–126. Springer (2022)
5. Boneh, D., Shoup, V.: A graduate course in applied cryptography. Draft 0.5 (2020)
6. Catalano, D., Giunta, E., Migliaro, F.: Anamorphic encryption: New constructions and homomorphic realizations. In: Advances in Cryptology – EUROCRYPT 2024. Lecture Notes in Computer Science, vol. 14652, pp. 33–62. Springer (2024). https://doi.org/10.1007/978-3-031-58723-8_2, https://doi.org/10.1007/978-3-031-58723-8_2
7. Choi, W., Collins, D., Liu, X., Stracovsky, R., Zikas, V.: A unified treatment of anamorphic encryption. In: Advances in Cryptology – CRYPTO 2026. Springer (2026), https://eprint.iacr.org/2025/309, full version available as Cryptology ePrint Archive, Paper 2025/309

8. Deo, A., Libert, B.: Anamorphic signatures with dictator and recipient unforgeability for long messages. In: Advances in Cryptology - ASIACRYPT 2025. Lecture Notes in Computer Science, vol. 16250, pp. 370–401. Springer (2025). https://doi.org/10.1007/978-981-95-5119-4_12, https://doi.org/10.1007/978-981-95-5119-4_12
9. Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the fiat-shamir transform. In: Galbraith, S.D., Nandi, M. (eds.) Progress in Cryptology - INDOCRYPT 2012. Lecture Notes in Computer Science, vol. 7668, pp. 60–79. Springer (2012). https://doi.org/10.1007/978-3-642-34931-7_5, https://doi.org/10.1007/978-3-642-34931-7_5
10. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO 1986. pp. 186–194. Springer (1986)
11. Jaeger, J., Stracovsky, R.: Dictators? friends? forgers. - breaking and fixing unforgeability definitions for anamorphic signature schemes. In: Advances in Cryptology - ASIACRYPT 2024. Lecture Notes in Computer Science, vol. 15485, pp. 105–137. Springer (2024). https://doi.org/10.1007/978-981-96-0888-1_4, https://doi.org/10.1007/978-981-96-0888-1_4
12. Krenn, S., Samelin, K., Slamanig, D.: Turning simulation into construction: New uses of NIZK simulators. Cryptology ePrint Archive, Paper 2025/2194 (2025)
13. Kutyłowski, M., Persiano, G., Phan, D.H., Yung, M., Zawada, M.: Anamorphic signatures: Secrecy from a dictator who only permits authentication! In: Advances in Cryptology – CRYPTO 2023. Lecture Notes in Computer Science, vol. 14082, pp. 759–790. Springer (2023). https://doi.org/10.1007/978-3-031-38545-2_25, https://doi.org/10.1007/978-3-031-38545-2_25
14. Libert, B., Peters, T., Joye, M., Yung, M.: Non-malleability from malleability: Simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. In: Advances in Cryptology - EUROCRYPT 2014. pp. 514–532. Springer (2014)
15. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing - STOC 1990. pp. 427–437. ACM (1990). https://doi.org/10.1145/100216.100273
16. Persiano, G., Phan, D.H., Yung, M.: Anamorphic encryption: Private communication against a dictator. In: Advances in Cryptology - EUROCRYPT 2022. Lecture Notes in Computer Science, vol. 13276, pp. 34–63. Springer (2022). https://doi.org/10.1007/978-3-031-07085-3_2, https://doi.org/10.1007/978-3-031-07085-3_2
17. Persiano, G., Phan, D.H., Yung, M.: Public-key anamorphism in (CCA-secure) public-key encryption and beyond. In: Advances in Cryptology – CRYPTO 2024. Lecture Notes in Computer Science, vol. 14921, pp. 422–455. Springer (2024). https://doi.org/10.1007/978-3-031-68379-4_13, https://doi.org/10.1007/978-3-031-68379-4_13
18. Pijnenburg, J., Poettering, B.: Key assignment schemes with authenticated encryption, revisited. IACR Transactions on Symmetric Cryptology pp. 40–67 (2020)
19. Rogaway, P.: Authenticated-encryption with associated-data. In: Proceedings of the 9th ACM Conference on Computer and Communications Security. pp. 98–107 (2002)

$\underline{\mathsf{G}^{\mathrm{ss}}_{\mathsf{NIZK},\mathcal{A},\mathcal{S}}(\lambda)}$:
$Q := \emptyset$, $(\mathsf{crs}, \tau) \leftarrow \mathcal{S}.\mathsf{Setup}(1^\lambda)$
$(x^*, \pi^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sim}}}(\mathsf{crs})$
If $(\mathsf{Verify}(x^*, \pi^*) = 1) \wedge ((x^*, \pi^*) \notin Q) \wedge (x^* \notin L_{\mathcal{R}})$: Return 1
Return 0

$\underline{\mathcal{O}_{\mathsf{Sim}}(x)}$:
$\pi \leftarrow \mathcal{S}.\mathsf{Prove}(\tau, x)$
$Q = Q \cup \{(x, \pi)\}$
Return π

Fig. 10: Simulation soundness game for NIZK

A Additional Preliminaries

Simulation soundness for NIZK. We now recall simulation soundness as an optional security property for NIZK.

Definition 18 (Simulation Soundness). *An NIZK scheme* NIZK *for* $\mathcal{R}$ *achieves simulation soundness, if for every PPT adversary* $\mathcal{A}$*, the advantage*

$$\mathbf{Adv}^{\mathrm{ss}}_{\mathsf{NIZK},\mathcal{A},\mathcal{S}}(\lambda) = \Pr[\mathsf{G}^{\mathrm{ss}}_{\mathsf{NIZK},\mathcal{A},\mathcal{S}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\mathrm{ss}}_{\mathsf{NIZK},\mathcal{A},\mathcal{S}}(\lambda)$ *is defined in Fig. 10, and* $\mathcal{S} = (\mathcal{S}.\mathsf{Setup}, \mathcal{S}.\mathsf{Prove})$ *is the simulator defined in Definition 4.*

The Fiat-Shamir heuristic [10]. Let $(\mathsf{P}, \mathsf{V}) = ((\mathsf{P}_1, \mathsf{P}_2), (\mathsf{V}_1, \mathsf{V}_2))$ be a Sigma protocol for a relation $\mathcal{R}$, where its conversations (a, c, z) belong to some space $\mathcal{CM} \times \mathcal{CL} \times \mathcal{Z}$. Let $\mathsf{Hash} : \mathcal{CM} \times \mathcal{X} \to \mathcal{CL}$ be a hash function. The Fiat-Shamir non-interactive proof system $\mathsf{NIZK}_{\mathrm{FS}} = (\mathsf{Prove}_{\mathrm{FS}}, \mathsf{Verify}_{\mathrm{FS}})$[8], with proof space $\mathcal{CM} \times \mathcal{Z}$, is as follows:

- $\mathsf{Prove}_{\mathrm{FS}}(x, w)$: On input $(x, w) \in \mathcal{X} \times \mathcal{W}$, this algorithm firstly generates $(a, \mathsf{st}) \leftarrow \mathsf{P}_1(x, w)$ and $c = \mathsf{Hash}(a, x)$, and then computes $z \leftarrow \mathsf{P}_2(a, c, x, w, \mathsf{st})$. Finally, it outputs $\pi = (a, z)$.
- $\mathsf{Verify}_{\mathrm{FS}}((a, z), x)$: On input $((a, z), x) \in (\mathcal{CM} \times \mathcal{Z}) \times \mathcal{X}$, this algorithm firstly computes $c = \mathsf{Hash}(a, x)$, and then runs $\mathsf{V}_2(x)$ to check whether (a, c, z) is a valid conversation for x. If so, $\mathsf{Verify}_{\mathrm{FS}}$ returns 1; otherwise, it returns 0.

According to [10,5], $\mathsf{NIZK}_{\mathrm{FS}}$ is an NIZK proof system if Hash is modeled as a random oracle.

General forking lemma. We next recall the general forking lemma, a standard tool for obtaining two related successful executions of an adversary under different random-oracle responses.

Lemma 1 (General Forking Lemma, cf. [3], Lemma 1). *Fix an integer* $q \geq 1$ *and a finite set* H *with* $|H| \geq 2$*. Let* IG *be a randomized algorithm called the input generator. Let* $\mathcal{A}$ *be a probabilistic algorithm that, on input a public parameter* x *(generated by* $x \leftarrow \mathsf{IG}$*) and* q *values* $h_1, \ldots, h_q$ *drawn independently and uniformly from* H*, outputs a pair* (J, σ)*. Here,* $J \in \{0, 1, \ldots, q\}$*, where* $J = 0$

[8] Note that we omit the setup algorithm here, as the FS transform generally does not involve any unusually complex setup computations.

indicates that $\mathcal{A}$ failed, and $J \geq 1$ indicates the index of the critical value upon which the successful output relies.

Let ϵ be the success probability of $\mathcal{A}$:

$$\epsilon := \Pr[x \leftarrow \mathsf{IG};\ h_1, \ldots, h_q \leftarrow H;\ (J, \sigma) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q) : J \geq 1].$$

There is a forking algorithm $\mathcal{F}_{\mathcal{A}}$ that, on input x, runs in the worst case in time at most twice the running time of $\mathcal{A}$ (a strict, not merely expected, polynomial-time bound), defined as follows: It picks random coins ρ for $\mathcal{A}$ and $h_1, \ldots, h_q \leftarrow H$, and runs $\mathcal{A}(x, h_1, \ldots, h_q; \rho)$ to get (J, σ). If $J = 0$, it halts with failure. Otherwise, it resamples fresh, independent, uniform values $h'_J, \ldots, h'_q \leftarrow H$, and reruns $\mathcal{A}$ with the exact same x, ρ and the modified array $h_1, \ldots, h_{J-1}, h'_J, \ldots, h'_q$, obtaining (J', σ'). It succeeds (outputs 1) if and only if $J = J'$ and $h_J \neq h'_J$.

Let ϵ_{fork} be the probability, over $x \leftarrow \mathsf{IG}$ and the coins of $\mathcal{F}_{\mathcal{A}}$, that $\mathcal{F}_{\mathcal{A}}$ succeeds. Then:

$$\epsilon_{\mathsf{fork}} \geq \epsilon \left(\frac{\epsilon}{q} - \frac{1}{|H|} \right).$$

Public Key Encryption The definition and security notions for public key encryption (PKE) are recalled as follows.

Definition 19 (PKE). *A Public Key Encryption scheme* $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *consists of three PPT algorithms:*

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$*: The key generation algorithm takes the security parameter λ as input, and then outputs a public key* pk *and a secret key* sk*.*
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}; s) \to \mathsf{ct}$*: The encryption algorithm takes a public key* pk*, a message* m*, and randomness $s \in \mathcal{RS}_{\mathsf{PKE.Enc}}$ as input, and then outputs a ciphertext $\mathsf{ct} \in \mathcal{CT}$.*
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{m}/\perp$*: The decryption algorithm takes a secret key* sk *and a string $\mathsf{ct} \in \mathcal{CT}$ as input, and then outputs a message* m *or a failure symbol $\perp$.*

Correctness. For any message m*, $s \in \mathcal{RS}_{\mathsf{PKE}}.\mathsf{Enc}$,*

$$\Pr\left[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda),\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}; s) :\ \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = \mathsf{m}\right] \geq 1 - \mathsf{negl}(\lambda).$$

The scheme is perfectly correct *if the probability above is equal to* 1*.*

Then, we recall the following three security properties:

Definition 20 (IND-CPA). *A PKE scheme* $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is IND-CPA secure if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage:*

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = |\Pr[\mathsf{G}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ is defined in Fig. 11.

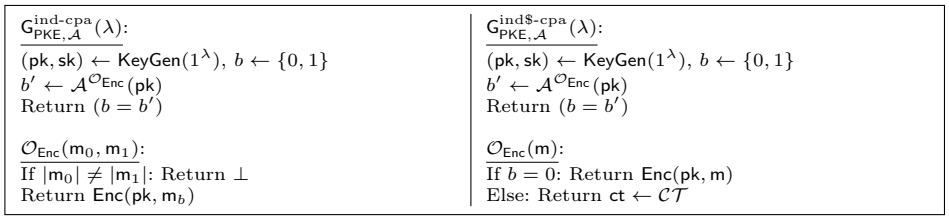

Fig. 11: IND-CPA and IND$-CPA security games for PKE

Definition 21 (IND$-CPA). *A PKE scheme* $\mathsf{PKE} = (\mathsf{KeyGen},\mathsf{Enc},\mathsf{Dec})$ *is IND$-CPA secure if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1,\mathcal{A}_2)$*, the advantage:*

$$\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = |\Pr[\mathsf{G}^{\text{ind\$-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\text{ind\$-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ *is defined in Fig. 11.*

Definition 22 (RCR). *A PKE scheme* $\mathsf{PKE} = (\mathsf{KeyGen},\mathsf{Enc},\mathsf{Dec})$ *achieves random ciphertext rejection, if for any* $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*, the probability that a uniformly random string* $\mathsf{ct} \leftarrow \mathcal{CT}$ *decrypts to a valid message is negligible:*

$$\Pr[\mathsf{Dec}(\mathsf{sk},\mathsf{ct}) \neq \perp : \mathsf{ct} \leftarrow \mathcal{CT}] \leq \mathsf{negl}(\lambda).$$

B Other Security Notions of ANIZK

Weak Robustness. Inspired by the robustness requirement for anamorphic encryption [2], we consider weak robustness for ANIZK. This property ensures that an honestly generated ordinary proof is not mistakenly interpreted as carrying an anamorphic message when it is processed using an independently generated double key.

Definition 23 (Weak Robustness). *An ANIZK scheme* ANIZK *is weakly robust, if for any PPT adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\text{wrob}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda) = \Pr[\mathsf{G}^{\text{wrob}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\text{wrob}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda)$ *is defined in Fig. 12.*

Recipient Anamorphic Simulation Soundness. This property requires that, even when given the double key and adaptive access to simulated anamorphic proofs, an adversary cannot output a fresh accepting statement-proof pair for a false statement, except with negligible probability.

Definition 24 (Recipient Anamorphic Simulation Soundness). *An ANIZK proof system* ANIZK*, with a PPT simulator* $\mathcal{S}_{\mathsf{ana}} = (\mathcal{S}_{\mathsf{ana}}.\mathsf{Setup},\mathcal{S}_{\mathsf{ana}}.\mathsf{aProve})$ *that*

$\underline{\mathsf{G}^{\mathrm{wrob}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda)}$:
$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{dk} \leftarrow \mathsf{aKeyGen}(1^\lambda)$, $(x, w) \leftarrow \mathcal{A}(\mathsf{crs})$
If $(x, w) \notin \mathcal{R}$: Return $\bot$
$\pi \leftarrow \mathsf{Prove}(x, w)$, $\mathsf{am} \leftarrow \mathsf{aExtract}(x, \pi, \mathsf{dk})$
Return 1 if and only if $\mathsf{am} \neq \bot$

Fig. 12: Recipient anamorphic weak robustness game for ANIZK

$\underline{\mathsf{G}^{\mathrm{rass}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda)}$:
$Q := \emptyset$, $(\mathsf{crs}, \tau) \leftarrow \mathcal{S}_{\mathsf{ana}}.\mathsf{Setup}(1^\lambda)$, $\mathsf{dk} \leftarrow \mathsf{aKeyGen}(1^\lambda)$
$(x^*, \pi^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sim}}}(\mathsf{crs}, \mathsf{dk})$
If $(\mathsf{Verify}(x^*, \pi^*) = 1) \wedge (x^* \notin \mathcal{L}_{\mathcal{R}}) \wedge ((x^*, \pi^*) \notin Q)$: Return 1
Return 0

$\underline{\mathcal{O}_{\mathsf{Sim}}(x, \mathsf{am})}$:
If $\mathsf{am} \notin \mathcal{AM}$: Return $\bot$
$\pi \leftarrow \mathcal{S}_{\mathsf{ana}}.\mathsf{aProve}(\tau, x, \mathsf{dk}, \mathsf{am})$, $Q := Q \cup \{(x, \pi)\}$
Return π

Fig. 13: Recipient anamorphic simulation soundness game for ANIZK

satisfies Definition 13, provides recipient anamorphic simulation soundness, if for any PPT adversary $\mathcal{A}$,

$$\mathbf{Adv}^{\mathrm{rass}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda) = \Pr[\mathsf{G}^{\mathrm{rass}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\mathrm{rass}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda)$ is defined in Fig. 13.

C Other proofs for Theorem 1

Since the proofs of recipient anamorphic zero knowledge and recipient anamorphic simulation extractability have already been given in the main text, we present here the remaining proofs, covering the ordinary NIZK security properties, anamorphic security, and strong robustness.

C.1 Proof of the ordinary completeness, zero knowledge, and computational soundness

Before proving the anamorphic properties, we show that the algorithms (Setup, $\mathsf{Prove}, \mathsf{Verify}$) of ANIZK in Fig. 6 consist of a standard NIZK proof system, i.e., they satisfy completeness, zero knowledge, and computational soundness as defined in Definition 4. These algorithms instantiate the salted Fiat-Shamir transform: the salt S is sampled uniformly from $\{0,1\}^{\ell(\lambda)}$, and the challenge is $c = \mathsf{H}(x, a, S)$. We work in the programmable random-oracle model, with $\mathsf{H} : \{0,1\}^* \to \mathcal{CL}$ and unambiguous encodings of (x, a, S). The parameters satisfy $2^{\ell(\lambda)} = \lambda^{\omega(1)}$ and $|\mathcal{CL}| = \lambda^{\omega(1)}$.

Completeness. The completeness is trivially guaranteed by the perfect completeness of the underlying Sigma protocol.

Zero Knowledge. We now prove the ordinary zero-knowledge property of ANIZK, where the hash function H is modeled as a random oracle. Let $\mathcal{S}_{\mathsf{Sigma}}$ be the special-HVZK simulator for the underlying Sigma protocol. We construct the NIZK simulator $\mathcal{S} = (\mathcal{S}.\mathsf{Setup}, \mathcal{S}.\mathsf{Prove})$ as follows:

- $\mathcal{S}.\mathsf{Setup}(1^\lambda)$: The setup simulator initializes an empty list L_{H} to simulate the random oracle H. The simulation state τ maintains L_{H} and grants the capability to program it. For any direct query (x, a, S) to H, the simulator returns $L_{\mathsf{H}}[x, a, S]$ if it is already defined; otherwise, it samples $c \leftarrow \mathcal{CL}$ uniformly at random, sets $L_{\mathsf{H}}[x, a, S] := c$, and returns c. It implicitly outputs crs as the access to this simulated random oracle.
- $\mathcal{S}.\mathsf{Prove}(\tau, x)$: On input a statement x, the proving simulator samples a challenge $c \leftarrow \mathcal{CL}$ and runs the Sigma protocol simulator $(a, z) \leftarrow \mathcal{S}_{\mathsf{Sigma}}(x, c)$. Then, it samples a random salt $S \leftarrow \{0,1\}^{\ell(\lambda)}$. If the entry $L_{\mathsf{H}}[x, a, S]$ is already defined, the simulator aborts. Otherwise, it programs the random oracle by setting $L_{\mathsf{H}}[x, a, S] := c$, and returns the proof $\pi = (a, S, z)$.

As specified in Fig. 2, the proof oracle $\mathcal{O}_{\mathsf{Sim}}$ ensures that $(x, w) \in \mathcal{R}$ before invoking $\mathcal{S}.\mathsf{Prove}(\tau, x)$, hence the simulator itself does not require the witness w.

Now, we turn to analyze the adversary's advantage.

Fix any PPT adversary $\mathcal{A}$. Let q_H and q_P be the upper bounds on the number of direct random oracle queries and valid proof queries made by $\mathcal{A}$, respectively. We proceed via a sequence of games. Let E_i denote the event that $\mathcal{A}$ outputs 1 in Game i.

Game 0: This is the real zero-knowledge game $\mathsf{G}^{\text{zk-real}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda)$. The random oracle H is simulated via lazy sampling, and the proofs are generated honestly by $\mathsf{Prove}(x, w)$. By definition, we have $\Pr[E_0] = \Pr[\mathsf{G}^{\text{zk-real}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda) = 1]$.

Game 1: This game is identical to Game 0, except that we modify the proof generation procedure to program the random oracle proactively. Specifically, for each valid proof query (x, w), the oracle computes $(a, \mathsf{st}) \leftarrow \mathsf{P}_1(x, w)$ and samples $S \leftarrow \{0,1\}^{\ell(\lambda)}$. Before proceeding, it checks if $L_{\mathsf{H}}[x, a, S]$ is already defined. If so, it aborts. Otherwise, it samples $c \leftarrow \mathcal{CL}$, sets $L_{\mathsf{H}}[x, a, S] := c$, computes $z \leftarrow \mathsf{P}_2(a, c, x, w, \mathsf{st})$, and returns $\pi = (a, S, z)$.

Game 1 proceeds identically to Game 0 unless the abort event occurs. Since S is sampled uniformly from $\{0,1\}^{\ell(\lambda)}$ *after* a is determined, and is completely independent of $\mathcal{A}$'s prior view, the probability of a collision with a previously queried tuple in L_{H} is bounded by the size of L_{H}. At the j-th proof query, L_{H} contains at most $q_H + j - 1$ entries. Taking a union bound over all q_P proof queries, we have

$$\big|\Pr[E_0] - \Pr[E_1]\big| \leq \sum_{j=1}^{q_P} \frac{q_H + j - 1}{2^{\ell(\lambda)}} = \frac{q_P q_H + q_P(q_P - 1)/2}{2^{\ell(\lambda)}}. \tag{4}$$

Game 2: This game modifies Game 1 by replacing the honest Sigma protocol prover with the special-HVZK simulator. For each proof query, it samples $c \leftarrow \mathcal{CL}$ and generates $(a, z) \leftarrow \mathcal{S}_{\mathsf{Sigma}}(x, c)$. It then samples $S \leftarrow \{0,1\}^{\ell(\lambda)}$, performs

the exact same abort check, programs $L_{\mathsf{H}}[x, a, S] := c$, and returns (a, S, z). Notice that Game 2 perfectly mirrors the simulated game $\mathsf{G}^{\text{zk-sim}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}}(\lambda)$. Thus, $\Pr[E_2] = \Pr[\mathsf{G}^{\text{zk-sim}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}}(\lambda) = 1]$.

To bound the difference between Game 1 and Game 2, we introduce intermediate hybrids Hyb_j for $0 \le j \le q_P$, where the first j valid proof queries are answered using the simulator $\mathcal{S}_{\mathsf{Sigma}}$ (as in Game 2), and the remaining queries are answered using the honest prover P (as in Game 1). Observe that Hyb_0 is exactly Game 1, and Hyb_{q_P} is exactly Game 2.

We construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the special-HVZK property of the underlying Sigma protocol to distinguish between these hybrids. $\mathcal{B}_1(1^\lambda)$ samples a random index $j^* \leftarrow \{1, \ldots, q_P\}$ and runs $\mathcal{A}$. It maintains L_{H} for direct random oracle queries. For the first $j^* - 1$ valid proof queries, it answers using $\mathcal{S}_{\mathsf{Sigma}}$ and RO programming. Upon receiving the j^*-th valid proof query (x^*, w^*), $\mathcal{B}_1$ outputs it as the challenge statement, passing its internal state to $\mathcal{B}_2$.

Upon receiving the challenge transcript (a^*, c^*, z^*) from the SHVZK challenger, $\mathcal{B}_2$ samples $S^* \leftarrow \{0,1\}^{\ell(\lambda)}$, applies the abort check on $L_{\mathsf{H}}[x^*, a^*, S^*]$, sets $L_{\mathsf{H}}[x^*, a^*, S^*] := c^*$, and returns $\pi^* = (a^*, S^*, z^*)$ to $\mathcal{A}$. For all subsequent proof queries, $\mathcal{B}_2$ answers them honestly using P (as in Game 1). Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

Let $b_{\mathsf{chal}} \in \{0,1\}$ be the hidden bit in the special-HVZK experiment. If $b_{\mathsf{chal}} = 0$ (real transcript), the simulated view perfectly matches Hyb_{j^*-1}; if $b_{\mathsf{chal}} = 1$ (simulated transcript), the view matches Hyb_{j^*}. Since j^* is chosen uniformly at random from $\{1, \ldots, q_P\}$, we have

$$\begin{aligned}
\mathbf{Adv}^{\text{shvzk}}_{\mathsf{Sigma},\mathcal{B},\mathcal{S}_{\mathsf{Sigma}}}(\lambda) &= \left|\Pr[\mathcal{B} \Rightarrow 1 \mid b_{\mathsf{chal}} = 0] - \Pr[\mathcal{B} \Rightarrow 1 \mid b_{\mathsf{chal}} = 1]\right| \\
&= \left|\sum_{j=1}^{q_P} \frac{1}{q_P} \Pr[\mathsf{Hyb}_{j-1} \Rightarrow 1] - \sum_{j=1}^{q_P} \frac{1}{q_P} \Pr[\mathsf{Hyb}_j \Rightarrow 1]\right| \\
&= \frac{1}{q_P} \left|\Pr[\mathsf{Hyb}_0 \Rightarrow 1] - \Pr[\mathsf{Hyb}_{q_P} \Rightarrow 1]\right| \\
&= \frac{1}{q_P} \left|\Pr[E_1] - \Pr[E_2]\right|.
\end{aligned}$$

Consequently, $\left|\Pr[E_1] - \Pr[E_2]\right| = q_P \cdot \mathbf{Adv}^{\text{shvzk}}_{\mathsf{Sigma},\mathcal{B},\mathcal{S}_{\mathsf{Sigma}}}(\lambda)$.

Combining the bounds, the overall zero-knowledge advantage is:

$$\begin{aligned}
\mathbf{Adv}^{\text{zk}}_{\mathsf{ANIZK},\mathcal{A},\mathcal{S}}(\lambda) &\le \left|\Pr[E_0] - \Pr[E_1]\right| + \left|\Pr[E_1] - \Pr[E_2]\right| \\
&\le \frac{q_P q_H + q_P(q_P - 1)/2}{2^{\ell(\lambda)}} + q_P \cdot \mathbf{Adv}^{\text{shvzk}}_{\mathsf{Sigma},\mathcal{B},\mathcal{S}_{\mathsf{Sigma}}}(\lambda) \le \mathsf{negl}(\lambda).
\end{aligned}$$

This concludes the proof.

Computational Soundness. Let $\mathcal{A}$ be any PPT adversary against the computational soundness of ANIZK, making at most q_H direct queries to the random oracle H.

The adversary $\mathcal{A}$ wins if it outputs a tuple $(x, \pi = (a, S, z))$ such that $x \notin \mathcal{L}_{\mathcal{R}}$ and $\mathsf{Verify}(x, \pi) = 1$. The verification algorithm accepts if and only if $\mathsf{V}_2(x, a, c, z) = 1$, where $c = \mathsf{H}(x, a, S)$.

For any statement $x \notin \mathcal{L}_{\mathcal{R}}$ and any commitment $a \in \mathcal{CM}$, we define

$$C_{x,a} = \{c \in \mathcal{CL} : \exists z \in \mathcal{Z} \text{ such that } \mathsf{V}_2(x, a, c, z) = 1\},$$

where $\mathcal{Z}$ is the response space of the underlying Sigma protocol. We claim that $|C_{x,a}| \leq 1$. Suppose towards a contradiction that $|C_{x,a}| \geq 2$. Then there exist two distinct challenges $c, c' \in C_{x,a}$ $(c \neq c')$ and corresponding responses z, z' such that both (a, c, z) and (a, c', z') are accepting transcripts for x. By the special soundness property of the underlying Sigma protocol, there exists an efficient deterministic extractor Ext that, on input (x, a, c, z, c', z'), outputs a valid witness w such that $(x, w) \in \mathcal{R}$. However, since $x \notin \mathcal{L}_{\mathcal{R}}$, no such witness can possibly exist. This contradiction implies that $|C_{x,a}| \leq 1$ unconditionally. Note that the set $C_{x,a}$ is uniquely determined by (x, a), and is completely independent of the random salt S and the output of the random oracle.

For the adversary $\mathcal{A}$ to succeed, it must be the case that the random oracle evaluates to a challenge $c = \mathsf{H}(x, a, S)$ such that $c \in C_{x,a}$. Since H is modeled as a random oracle, for every fresh query tuple (x, a, S), the oracle returns a challenge c chosen uniformly at random from $\mathcal{CL}$. Thus,

$$\Pr[c \in C_{x,a}] \leq \frac{|C_{x,a}|}{|\mathcal{CL}|} \leq \frac{1}{|\mathcal{CL}|}.$$

The adversary $\mathcal{A}$ makes at most q_H direct queries to the random oracle. Furthermore, if the final proof $\pi = (a, S, z)$ output by $\mathcal{A}$ involves a tuple (x, a, S) that was never queried directly, Verify will make one additional implicit evaluation of $\mathsf{H}(x, a, S)$, which acts as a fresh independent query. Applying a union bound over all (at most $q_H + 1$) distinct queries to H, we obtain

$$\begin{aligned}\Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda),\ (x, \pi) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{H}}}(\mathsf{crs}) : \mathsf{Verify}(x, \pi) = 1 \wedge x \notin \mathcal{L}_{\mathcal{R}}\right] \\ \leq \frac{q_H + 1}{|\mathcal{CL}|} \leq \mathsf{negl}(\lambda).\end{aligned}$$

This concludes the proof of computational soundness.

C.2 Proof of Anamorphic Security

Proof. We construct the Equiv algorithm in Fig. 14 and show its relation with other algorithms.

We prove the anamorphic security of ANIZK using a sequence of games. For $i \in \{0, 1\}$, let E_i denote the event that the output of Game i is 1.

Let $\mathcal{A}$ be a PPT adversary in the anamorphic security game $\mathsf{G}^{\mathsf{anam}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda)$ of Fig. 4. We first verify that the algorithm Equiv in Fig. 14 produces a valid opening of the anamorphic proof as an ordinary proof. Write

$$\mathcal{RS}_{\mathsf{Prove}} = \mathcal{RS}_{\mathsf{Sigma}} \times \{0, 1\}^{\ell(\lambda)} \quad \text{and} \quad \mathcal{RS}_{\mathsf{aProve}} = \mathcal{RS}_{\mathsf{Sigma}} \times \mathcal{RS}_{\mathsf{AEAD}}.$$

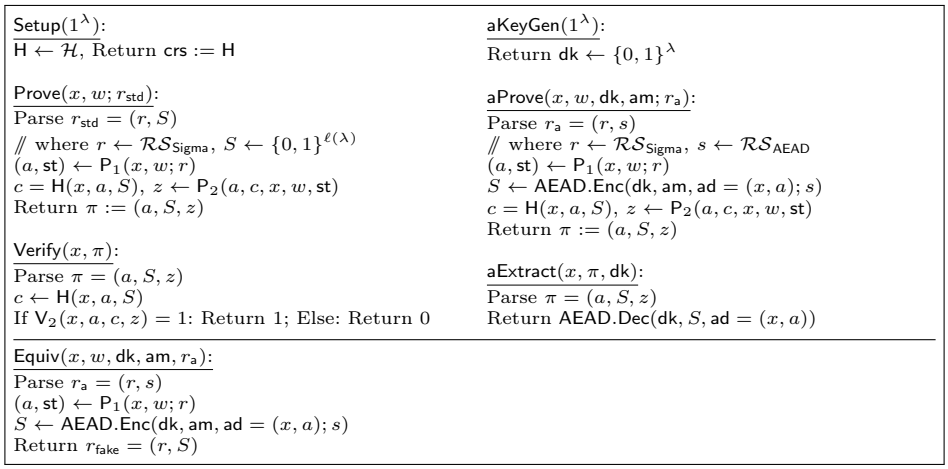

Fig. 14: The generic construction of ANIZK

Fix any valid $(x, w) \in \mathcal{R}$, any $\mathsf{am} \in \mathcal{AM}$, any double key dk, and any anamorphic randomness $r_{\mathsf{a}} = (r, s)$. Both aProve and Equiv compute

$$(a, \mathsf{st}) \leftarrow \mathsf{P}_1(x, w; r), \qquad S \leftarrow \mathsf{AEAD.Enc}(\mathsf{dk}, \mathsf{am}, \mathsf{ad} = (x, a); s).$$

Consequently, given $r_{\mathsf{fake}} = (r, S)$ returned by Equiv, running Prove with r_{fake} for (x, w), recomputes the same $a \leftarrow \mathsf{P}_1(x, w; r)$, the same challenge $c = \mathsf{H}(x, a, S)$, and the same response $z = \mathsf{P}_2(a, c, x, w, \mathsf{st})$. Hence

$$\mathsf{Prove}\big(x, w; \mathsf{Equiv}(x, w, \mathsf{dk}, \mathsf{am}, r_{\mathsf{a}})\big) = \mathsf{aProve}(x, w, \mathsf{dk}, \mathsf{am}; r_{\mathsf{a}}). \tag{5}$$

Thus, the π generated by $\mathsf{Prove}(x, w; r_{\mathsf{fake}})$ is always consistent in the anamorphic branch.

We now compare the following two games. For $i \in \{0, 1\}$, let E_i denote the event that $\mathcal{A}$ outputs 1 in Game i.

Game 0. The challenger generates $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ and $\mathsf{dk} \leftarrow \mathsf{aKeyGen}(1^\lambda)$, then sends crs to $\mathcal{A}$. The challenger answers $\mathcal{A}$'s query as follows.

- $\mathcal{O}_{\mathsf{chal}}(x, w, \mathsf{am})$: If $(x, w) \notin \mathcal{R}$ or $\mathsf{am} \notin \mathcal{AM}$, the challenger returns $\perp$. Otherwise, it samples $r \leftarrow \mathcal{RS}_{\mathsf{Sigma}}$, $S \leftarrow \{0,1\}^{\ell(\lambda)}$, sets $r^* = (r, S)$, computes $\pi \leftarrow \mathsf{Prove}(x, w; r^*)$, and gives (π, r^*) to $\mathcal{A}$.

This is exactly the $b = 0$ branch of $\mathsf{G}^{\mathrm{anam}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda)$.

Game 1. Game 1 is the same as Game 0, except that the challenger answers $\mathcal{A}$'s query as follows.

- $\mathcal{O}_{\mathsf{chal}}(x, w, \mathsf{am})$: If $(x, w) \notin \mathcal{R}$ or $\mathsf{am} \notin \mathcal{AM}$, the challenger returns $\perp$. Otherwise, it samples $r_{\mathsf{a}} = (r, s) \leftarrow \mathcal{RS}_{\mathsf{aProve}}$, computes $\pi \leftarrow \mathsf{aProve}(x, w, \mathsf{dk}, \mathsf{am}; r_a)$, $r_{\mathsf{fake}} \leftarrow \mathsf{Equiv}(x, w, \mathsf{dk}, \mathsf{am}, r_{\mathsf{a}})$, sets $r^* = r_{\mathsf{fake}}$ and gives (π, r^*) to $\mathcal{A}$.

This is exactly the $b = 1$ branch of $\mathsf{G}^{\mathrm{anam}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda)$.

By the definition of Game 0 and Game 1,

$$\begin{aligned}\mathbf{Adv}^{\mathrm{anam}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda) &= |\Pr[\mathsf{G}^{\mathrm{anam}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \\ &\leq |\frac{1}{2}(1 - \Pr[E_0]) + \frac{1}{2}\Pr[E_1] - \frac{1}{2}| \\ &\leq \frac{1}{2}|\Pr[E_1] - \Pr[E_0]|. \end{aligned} \tag{6}$$

We bound this difference by constructing a PPT adversary $\mathcal{B}$ against the IND\$-CPA security of AEAD.

$\mathcal{B}$ runs $\mathsf{Setup}(1^\lambda)$ to generate crs and runs $\mathsf{aKeyGen}(1^\lambda)$ to generate dk, then it sends crs to $\mathcal{A}$. $\mathcal{B}$ answers $\mathcal{A}$'s query as follows

- $\underline{\mathcal{O}_{\mathsf{chal}}(x, w, \mathsf{am})}$: If $(x, w) \notin \mathcal{R}$ or $\mathsf{am} \notin \mathcal{AM}$, $\mathcal{B}$ returns $\perp$. Otherwise, it samples $r \leftarrow \mathcal{RS}_{\mathsf{Sigma}}$, computes $(a, \mathsf{st}) \leftarrow \mathsf{P}_1(x, w; r)$, and submits $(\mathsf{am}, \mathsf{ad} = (x, a))$ to its IND\$-CPA encryption oracle. Let S be the returned string. Adversary $\mathcal{B}$ computes

$$c = \mathsf{H}(x, a, S), z \leftarrow \mathsf{P}_2(a, c, x, w, \mathsf{st}), \pi = (a, S, z), r^* = (r, S)$$

  and sends (π, r^*) to $\mathcal{A}$.

Finally, when $\mathcal{B}$ receives the final output β from $\mathcal{A}$, it returns $b' = 1 - \beta$ to the challenger of the IND\$-CPA game.

If the IND\$-CPA hidden bit $b_\$$ is 0, then S is an encryption of am under associated data (x, a). By Eq. (5), the view of $\mathcal{A}$ is distributed exactly as in Game 1. If $b_\$ = 1$, then S is an independent uniform string in $\{0,1\}^{\ell(\lambda)}$, and the view of $\mathcal{A}$ is distributed exactly as in Game 0. Hence,

$$\begin{aligned}\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathsf{AEAD},\mathcal{B}}(\lambda) &= |\Pr[\mathsf{G}^{\text{ind\$-cpa}}_{\mathsf{AEAD},\mathcal{B}}(\lambda) = 1] - \frac{1}{2}| \\ &\geq |\frac{1}{2}\Pr[E_1] + \frac{1}{2}(1 - \Pr[E_0]) - \frac{1}{2}| \\ &\geq \frac{1}{2}|\Pr[E_1] - \Pr[E_0]|. \end{aligned} \tag{7}$$

Combining this equality with Eq. (6)-(7), we obtain

$$\mathbf{Adv}^{\mathrm{anam}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathsf{AEAD},\mathcal{B}}(\lambda) \leq \mathsf{negl}(\lambda).$$

Therefore, the proposed ANIZK scheme satisfies anamorphic security.

C.3 Proof of Strong Robustness

Proof. We prove the strong robustness of ANIZK using a sequence of games. For $i \in \{0, 1\}$, let E_i denote the event that the output of Game i is 1.

Game 0. This is the original strong robustness game $\mathsf{G}^{\mathrm{srob}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda)$. Hence,

$$\mathbf{Adv}^{\mathrm{srob}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda) = \Pr[\mathsf{G}^{\mathrm{srob}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda) = 1] = \Pr[E_0].$$

Game 1. Game 1 is the same as Game 0, except that the winning condition additionally requires that for $\mathcal{A}$'s final output $(x^*, \pi^* = (a^*, S^*, z^*))$, it holds that

$$(S^*, (x^*, a^*)) \notin Q_{\mathsf{AEAD}},$$

where $Q_{\mathsf{AEAD}} := \{(S_i, (x_i, a_i)) : \exists z_i \text{ such that } (x_i, \pi_i = (a_i, S_i, z_i)) \in Q\}$.

Let Bad_1 be the event that $\mathcal{A}$ wins Game 0 but the $(S^*, (x^*, a^*))$ belongs to Q_{AEAD}. The two games are identical unless Bad_1 occurs. Therefore,

$$\Pr[E_0] \leq \Pr[E_1] + \Pr[\mathsf{Bad}_1] \tag{8}$$

We first bound $\Pr[\mathsf{Bad}_1]$ by constructing a PPT adversary $\mathcal{B}_1$ against the quasi-unique responses property of the Sigma protocol.

Adversary $\mathcal{B}_1$ runs $\mathsf{Setup}(1^\lambda)$ and $\mathsf{aKeyGen}(1^\lambda)$ to generate crs and dk, then sends crs to $\mathcal{A}$, and answers anamorphic-proving queries exactly as in Game 0. In particular,

- $\mathcal{O}_{\mathsf{aProve}^*}(x_i, w_i, \mathsf{am}_i)$: If $(x_i, w_i) \notin \mathcal{R}$ or $\mathsf{am}_i \notin \mathcal{AM}$, $\mathcal{B}_1$ returns $\perp$. Otherwise, $\mathcal{B}_1$ computes $(a_i, \mathsf{st}_i) \leftarrow \mathsf{P}_1(x_i, w_i)$ and $S_i \leftarrow \mathsf{AEAD.Enc}(\mathsf{dk}, \mathsf{am}_i, \mathsf{ad} = (x_i, a_i))$. Then $\mathcal{B}_1$ computes $c_i = \mathsf{H}(x_i, a_i, S_i)$, $z_i \leftarrow \mathsf{P}_2(a_i, c_i, x_i, w_i, \mathsf{st}_i)$, and returns $\pi_i = (a_i, S_i, z_i)$ to $\mathcal{A}$ and sets $Q = Q \cup \{(x_i, \pi_i)\}$ and $Q_{\mathsf{AEAD}} = Q_{\mathsf{AEAD}} \cup \{(S_i, (x_i, a_i))\}$.

Suppose that Bad_1 occurs. Then, for some previous oracle response $\pi_i = (a_i, S_i, z_i)$, we have

$$(S^*, (x^*, a^*)) = (S_i, (x_i, a_i)).$$

Consequently, $x^* = x_i$, $a^* = a_i$ and $S^* = S_i$.

Since $\mathcal{A}$ wins the game, the final proof $(x^*, \pi^* = (a^*, S^*, z^*))$ is accepting. Besides, the $(x_i, \pi_i = (a_i, S_i, z_i))$ generated by $\mathcal{O}_{\mathsf{aProve}}$, also can be accepted. Moreover, the winning condition of $\mathcal{A}$ requires $(x^*, \pi^*) \notin Q$, which suggests $z^* \neq z_i$. Therefore, $\mathcal{B}_1$ outputs $(x_i, a_i, c_i, z_i, z^*)$, which consists of two different accepting responses for the fixed tuple (x_i, a_i, c_i). Thus,

$$\Pr[\mathsf{Bad}_1] \leq \mathbf{Adv}^{\text{qur-uni}}_{\mathsf{Sigma},\mathcal{B}_1}(\lambda). \tag{9}$$

We then bound $\Pr[E_1]$ by constructing a PPT adversary $\mathcal{B}_2$ against the INT-CTXT security of AEAD, using the adversary $\mathcal{A}$ from Game 1. Adversary $\mathcal{B}_2$ runs $\mathsf{Setup}(1^\lambda)$ to generate crs and sends it to $\mathcal{A}$.

$\mathcal{B}_2$ answers $\mathcal{A}$'s query as follows:

- $\mathcal{O}_{\mathsf{aProve}^*}(x_i, w_i, \mathsf{am}_i)$: If $(x_i, w_i) \notin \mathcal{R}$ or $\mathsf{am}_i \notin \mathcal{AM}$, $\mathcal{B}_2$ returns $\perp$. Otherwise, $\mathcal{B}_2$ computes $(a_i, \mathsf{st}_i) \leftarrow \mathsf{P}_1(x_i, w_i)$ and sends $(\mathsf{am}_i, \mathsf{ad}_i = (x_i, a_i))$ to its INT-CTXT encryption oracle. Let S_i be the returned string. Then $\mathcal{B}_2$ computes $c_i = \mathsf{H}(x_i, a_i, S_i)$, $z_i \leftarrow \mathsf{P}_2(a_i, c_i, x_i, w_i, \mathsf{st}_i)$, and returns $\pi_i = (a_i, S_i, z_i)$ to $\mathcal{A}$ and sets $Q = Q \cup \{(x_i, \pi_i)\}$ and $Q_{\mathsf{AEAD}} = Q_{\mathsf{AEAD}} \cup \{(S_i, (x_i, a_i))\}$.

These responses have exactly the distribution of the anamorphic-proving oracle in Game 1.

When $\mathcal{A}$ outputs (x^*, π^*), adversary $\mathcal{B}_2$ parse $\pi^* = (a^*, S^*, z^*)$ and returns $(S^*, \mathsf{ad}^* = (x^*, a^*))$ as its INT-CTXT forgery. When $\mathcal{A}$ wins Game 1, its final output pair $(x^*, \pi^* = (a^*, S^*, z^*))$ is not in Q and $\mathsf{aExtract}(x^*, \pi^*, \mathsf{dk}) \neq \perp$ (i.e., $\mathsf{AEAD.Dec}(\mathsf{dk}, S^*, \mathsf{ad}^*) \neq \perp$).

Hence, $\mathcal{B}_2$'s final output $(S^*, \mathsf{ad}^* = (x^*, a^*))$ satisfies $\mathsf{AEAD.Dec}(\mathsf{dk}, S^*, \mathsf{ad}^*) \neq \perp$ and $(S^*, \mathsf{ad}^* = (x^*, a^*)) \notin Q_{\mathsf{AEAD}}$.

So $\mathcal{B}_2$ wins its INT-CTXT game whenever $\mathcal{A}$ wins Game 1, and

$$\Pr[E_1] \leq \mathbf{Adv}^{\text{int-ctxt}}_{\mathsf{AEAD},\mathcal{B}_2}(\lambda). \tag{10}$$

Combining Eqs. (8) – (10), we obtain

$$\mathbf{Adv}^{\text{srob}}_{\mathsf{ANIZK},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\text{qur-uni}}_{\mathsf{Sigma},\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\text{int-ctxt}}_{\mathsf{AEAD},\mathcal{B}_2}(\lambda) \leq \mathsf{negl}(\lambda).$$

Therefore, the proposed ANIZK scheme satisfies strong robustness.

D Security notions of recipient anamorphic encryption

Adapted from [7], we now recall the security notions for RAME: STE-xCCA1, SIND-aCCA1, and SIND-nCCA.

Definition 25 (STE-xCCA1). *A RAME scheme* $\mathsf{RAME} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{aKeyGen}, \mathsf{aEnc}, \mathsf{aDec})$ *is STE-xCCA1 secure, if for every PPT adversary* $\mathcal{A}$*, the advantage*

$$\mathbf{Adv}^{\text{STE-xCCA1}}_{\mathsf{RAME},\mathcal{A}}(\lambda) := |\Pr[\mathsf{G}^{\text{STE-xCCA1}}_{\mathsf{RAME},\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{G}^{\text{STE-xCCA1}}_{\mathsf{RAME},\mathcal{A},1}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\text{STE-xCCA1}}_{\mathsf{RAME},\mathcal{A},0}(\lambda)$ *and* $\mathsf{G}^{\text{STE-xCCA1}}_{\mathsf{RAME},\mathcal{A},1}(\lambda)$ *are defined in Fig.15.*

Definition 26 (SIND-aCCA1). *A RAME scheme* $\mathsf{RAME} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{aKeyGen}, \mathsf{aEnc}, \mathsf{aDec})$ *is SIND-aCCA1 secure, if for every PPT adversary* $\mathcal{A}$*, the advantage*

$$\mathbf{Adv}^{\text{SIND-aCCA1}}_{\mathsf{RAME},\mathcal{A}}(\lambda) := |\Pr[\mathsf{G}^{\text{SIND-aCCA1}}_{\mathsf{RAME},\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{G}^{\text{SIND-aCCA1}}_{\mathsf{RAME},\mathcal{A},1}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\text{SIND-aCCA1}}_{\mathsf{RAME},\mathcal{A},0}(\lambda)$ *and* $\mathsf{G}^{\text{SIND-aCCA1}}_{\mathsf{RAME},\mathcal{A},1}(\lambda)$ *are defined in Fig.15.*

Definition 27 (SIND-nCCA). *A RAME encryption scheme* $\mathsf{RAME} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{aKeyGen}, \mathsf{aEnc}, \mathsf{aDec})$ *is SIND-nCCA secure, if for every PPT adversary* $\mathcal{A}$*, the advantage*

$$\mathbf{Adv}^{\text{SIND-nCCA}}_{\mathsf{RAME},\mathcal{A}}(\lambda) := |\Pr[\mathsf{G}^{\text{SIND-nCCA}}_{\mathsf{RAME},\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{G}^{\text{SIND-nCCA}}_{\mathsf{RAME},\mathcal{A},1}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\text{SIND-nCCA}}_{\mathsf{RAME},\mathcal{A},0}(\lambda)$ *and* $\mathsf{G}^{\text{SIND-nCCA}}_{\mathsf{RAME},\mathcal{A},1}(\lambda)$ *are defined in Fig.15.*

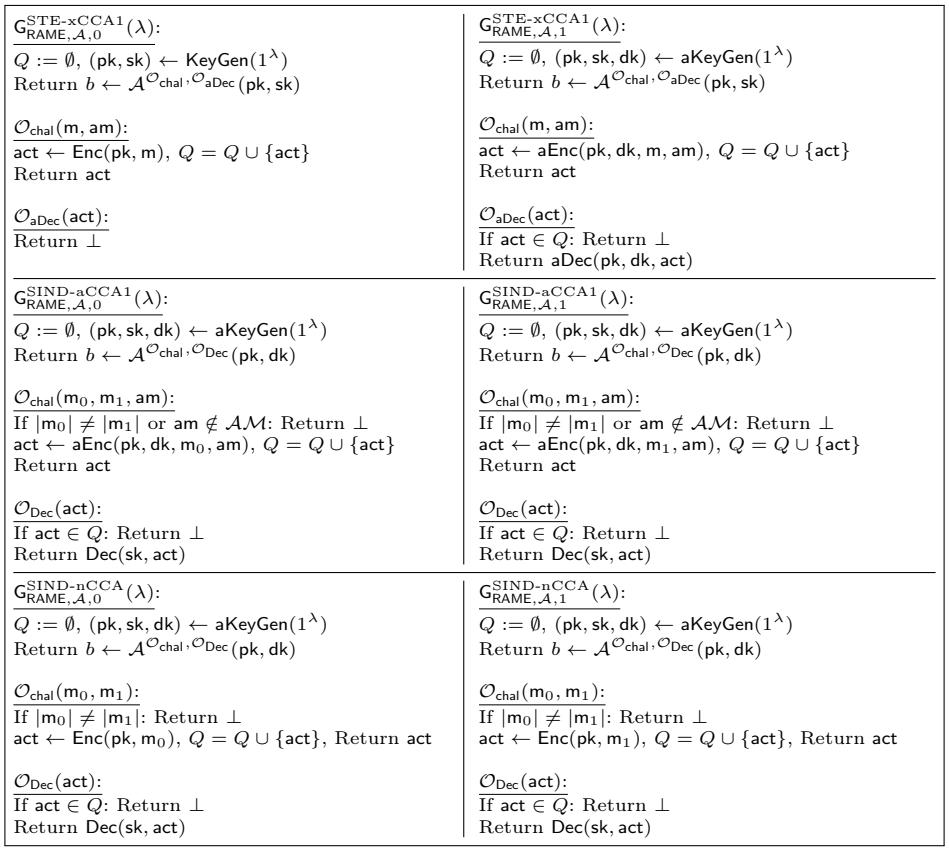

Fig. 15: STE-xCCA1, SIND-aCCA1, and SIND-nCCA security games for a symmetric anamorphic encryption scheme RAME

E Proof of Theorem 2

We prove the Theorem 2 by establishing STE-xCCA1, SIND-aCCA1, and SIND-nCCA security.

E.1 Proof of STE-xCCA1

Proof. We use the following sequence of games. For $i \in \{0, 1, 2, 3\}$, let E_i denote the event that the output of Game i is 1.

Game 0. This is the original game $\mathsf{G}^{\text{STE-xCCA1}}_{\mathsf{RAME},\mathcal{A},0}(\lambda)$. The challenge oracle returns normal ciphertexts, and $\mathcal{O}_{\mathsf{aDec}}$ always returns $\perp$. Thus,

$$\Pr[E_0] = \Pr[\mathsf{G}^{\text{STE-xCCA1}}_{\mathsf{RAME},\mathcal{A},0}(\lambda) = 1]. \tag{11}$$

Game 1. Game 1 is the same as Game 0, except that the challenger additionally generates $\mathsf{dk} \leftarrow \mathsf{ANIZK.aKeyGen}(1^\lambda)$ and keeps it hidden from $\mathcal{A}$. The public

and normal secret keys have the same distribution under $\mathsf{RAME.KeyGen}$ and $\mathsf{RAME.aKeyGen}$, while dk is independent and unused. Therefore,

$$\Pr[E_0] = \Pr[E_1]. \tag{12}$$

Game 2. Game 2 is the same as Game 1, except that the proof in every challenge ciphertext is changed from

$$\mathsf{ANIZK.Prove}(x, (\mathsf{m}, r_1, r_2)) \quad \text{to} \quad \mathsf{ANIZK.aProve}(x, (\mathsf{m}, r_1, r_2), \mathsf{dk}, \mathsf{am}).$$

Both PKE components remain unchanged, and $\mathcal{O}_{\mathsf{aDec}}$ still always returns $\perp$.

To bound this hop, construct a PPT anamorphic security adversary $\mathcal{B}_1$ for ANIZK. Let $b_{\$}$ be a hidden bit of that game.

After receiving crs from its challenger, $\mathcal{B}_1$ generates two PKE key pairs $(\mathsf{pk}_1, \mathsf{sk}_1)$ and $(\mathsf{pk}_2, \mathsf{sk}_2)$. Then, $\mathcal{B}_1$ sends $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$ and $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$ to $\mathcal{A}$. After that, $\mathcal{B}_1$ answers the $\mathcal{A}$'s query as follows:

- $\mathcal{O}_{\mathsf{chal}}(\mathsf{m}, \mathsf{am})$: $\mathcal{B}_1$ samples two PKE encryption randomness $(r_1, r_2) \leftarrow \mathcal{RS}^2_{\mathsf{PKE.Enc}}$ and, for every challenge query $(\mathsf{m}, \mathsf{am})$, computes $\mathsf{ct}_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_1, \mathsf{m}; r_1)$ and $\mathsf{ct}_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, \mathsf{m}; r_2)$. Then $\mathcal{B}_1$ sends $(x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2), w = (\mathsf{m}, r_1, r_2), \mathsf{am})$ to its own challenging oracle of the ANIZK anamorphic security game. Assume that (π, r^*) is the response of the challenging oracle. $\mathcal{B}_1$ sets $\mathsf{act} := (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ and $Q = Q \cup \{\mathsf{act}\}$. Finally, $\mathcal{B}_1$ returns act to $\mathcal{A}$.

Finally, $\mathcal{B}_1$ returns the final output of $\mathcal{A}$.

$\mathcal{B}_1$'s simulation is Game 1 when the hidden $b_{\$} = 0$ of the challenging oracle, and Game 2 when the hidden $b_{\$} = 1$ of the challenging oracle. Hence,

$$|\Pr[E_2] - \Pr[E_1]| \leq 2 \cdot \mathbf{Adv}^{\mathrm{anam}}_{\mathsf{ANIZK}, \mathcal{B}_1}(\lambda). \tag{13}$$

Game 3. Game 3 is the same as Game 2, except that $\mathcal{O}_{\mathsf{aDec}}$ now behaves as in $\mathsf{G}^{\text{STE-xCCA1}}_{\mathsf{RAME}, \mathcal{A}, 1}(\lambda)$: it returns $\perp$ on every ciphertext in the challenge set Q and otherwise runs $\mathsf{RAME.aDec}$. Therefore, Game 3 is exactly $\mathsf{G}^{\text{STE-xCCA1}}_{\mathsf{RAME}, \mathcal{A}, 1}(\lambda)$, and

$$\Pr[E_3] = \Pr[\mathsf{G}^{\text{STE-xCCA1}}_{\mathsf{RAME}, \mathcal{A}, 1}(\lambda) = 1]. \tag{14}$$

It remains to bound the transition from Game 2 to Game 3. Let Bad_1 be the event that $\mathcal{A}$ submits a non-challenge ciphertext $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi) \notin Q$ such that, for $x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$,

$$\mathsf{ANIZK.Verify}(x, \pi) = 1 \quad \text{and} \quad \mathsf{ANIZK.aExtract}(x, \pi, \mathsf{dk}) \neq \perp.$$

The games are identical until this event occurs, so

$$|\Pr[E_3] - \Pr[E_2]| \leq \Pr[\mathsf{Bad}_1]. \tag{15}$$

To bound $\Pr[\mathsf{Bad}_1]$, we construct an adversary $\mathcal{B}_2$ against ANIZK strong robustness.

After receiving crs from the challenger of the game of strong robustness of ANIZK, $\mathcal{B}_2$ generates two PKE key pairs $(\mathsf{pk}_1, \mathsf{sk}_1)$ and $(\mathsf{pk}_2, \mathsf{sk}_2)$. Then, $\mathcal{B}_2$ sends $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$ and $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$ to $\mathcal{A}$.

After that, when $\mathcal{A}$ queries the challenge proof oracle $\mathcal{O}_{\mathsf{chal}}$, $\mathcal{B}_2$ answers it as follows:

– $\underline{\mathcal{O}_{\mathsf{chal}}(\mathsf{m}, \mathsf{am})}$: $\mathcal{B}_2$ chooses two randomness $(r_1, r_2) \leftarrow \mathcal{RS}^2_{\mathsf{PKE.Enc}}$, computes $\underline{\mathsf{ct}_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_1, \mathsf{m}; r_1)}$ and $\mathsf{ct}_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, \mathsf{m}; r_2)$. Then $\mathcal{B}_2$ sends $(x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2), w = (\mathsf{m}, r_1, r_2), \mathsf{am})$ to its own oracle $\mathcal{O}_{\mathsf{aProve}^*}$ of the ANIZK strong robustness game. Assume π is the response of $\mathcal{O}_{\mathsf{aProve}^*}$. $\mathcal{B}_2$ sets $\mathsf{act} := (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ and $Q = Q \cup \{\mathsf{act}\}$. Finally, $\mathcal{B}_2$ returns act to $\mathcal{A}$.

When event Bad_1 occurs, $\mathcal{A}$ queries $\mathsf{act} = (\mathsf{ct}_1^*, \mathsf{ct}_2^*, \pi^*) \notin Q$ to $\mathcal{O}_{\mathsf{aDec}}$, and it satisfies for $x^* = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1^*, \mathsf{ct}_2^*)$,

$$\mathsf{ANIZK.Verify}(x^*, \pi^*) = 1 \quad \text{and} \quad \mathsf{ANIZK.aExtract}(x^*, \pi^*, \mathsf{dk}) \neq \bot.$$

Then $\mathcal{B}_2$ aborts and outputs (x^*, π^*) as its final output in the ANIZK strong robustness game. Since $\mathsf{act} = (\mathsf{ct}_1^*, \mathsf{ct}_2^*, \pi^*) \notin Q$, (x^*, π^*) is not recorded in $\mathcal{O}_{\mathsf{aProve}^*}$ of the ANIZK strong robustness game. Therefore,

$$\Pr[\mathsf{Bad}_1] \leq \mathbf{Adv}^{\mathrm{srob}}_{\mathsf{ANIZK},\mathcal{B}_2}(\lambda). \tag{16}$$

Combining Eqs. (11)–(16) and applying the triangle inequality, we obtain

$$\mathbf{Adv}^{\text{STE-xCCA1}}_{\mathsf{RAME},\mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\mathrm{anam}}_{\mathsf{ANIZK},\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\mathrm{srob}}_{\mathsf{ANIZK},\mathcal{B}_2}(\lambda) \leq \mathsf{negl}(\lambda).$$

Thus, RAME satisfies STE-xCCA1 security.

E.2 Proof of SIND-aCCA1

Proof. Let $\mathcal{S}_{\mathsf{ana}} = (\mathcal{S}_{\mathsf{ana}}.\mathsf{Setup}, \mathcal{S}_{\mathsf{ana}}.\mathsf{aProve})$ be the recipient anamorphic simulator in the recipient anamorphic zero knowledge security of ANIZK. For $i \in \{0, 1, 2, 3, 4, 5, 6\}$, let E_i denote the event that the output of Game i is 1.

Game 0. This is the original game $\mathsf{G}^{\text{SIND-aCCA1}}_{\mathsf{RAME},\mathcal{A},0}(\lambda)$. Thus, both PKE components of each challenge encrypt m_0, and the attached proof is an honest anamorphic proof concealing am. Hence,

$$\Pr[E_0] = \Pr[\mathsf{G}^{\text{SIND-aCCA1}}_{\mathsf{RAME},\mathcal{A},0}(\lambda) = 1]. \tag{17}$$

Game 1. Game 1 is the same as Game 0, except that the challenger uses $(\mathsf{crs}, \tau) \leftarrow \mathcal{S}_{\mathsf{ana}}.\mathsf{Setup}(1^\lambda)$ and replaces every honest challenge proof by a proof generated with $\mathcal{S}_{\mathsf{ana}}.\mathsf{aProve}(\tau, x, \mathsf{dk}, \mathsf{am})$.

To bound this hop, we construct a PPT adversary $\mathcal{B}_1$ in the recipient anamorphic zero knowledge game in Definition 13.

After $\mathcal{B}_1$ receives the $(\mathsf{crs}, \mathsf{dk})$ from its challenger, it generates two PKE pairs $(\mathsf{pk}_1, \mathsf{sk}_1)$, $(\mathsf{pk}_2, \mathsf{sk}_2)$, sets $\mathsf{pk} := (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$, $\mathsf{sk} := (\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$, and sends $(\mathsf{pk}, \mathsf{dk})$ to $\mathcal{A}$. When $\mathcal{A}$ queries the oracles $\mathcal{O}_{\mathsf{chal}}$ or $\mathcal{O}_{\mathsf{Dec}}$, $\mathcal{B}_1$ answers $\mathcal{A}$ as follows:

– $\underline{\mathcal{O}_{\mathsf{chal}}(\mathsf{m}_0, \mathsf{m}_1, \mathsf{am}):}$ If $|\mathsf{m}_0| \neq |\mathsf{m}_1|$ or $\mathsf{am} \notin \mathcal{AM}$, $\mathcal{B}_1$ returns $\perp$. Otherwise, it chooses two randomness $(r_1, r_2) \leftarrow \mathcal{RS}^2_{\mathsf{PKE.Enc}}$, computes $\mathsf{ct}_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_1, \mathsf{m}_0; r_1)$ and $\mathsf{ct}_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, \mathsf{m}_0; r_2)$. Then $\mathcal{B}_1$ sends $(x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2), w = (\mathsf{m}_0, r_1, r_2), \mathsf{am})$ to its own proving oracle in the recipient anamorphic zero-knowledge game. Assume π is the response of the proving oracle. $\mathcal{B}_1$ sets $\mathsf{act} := (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ and $Q = Q \cup \{\mathsf{act}\}$. Finally, $\mathcal{B}_1$ returns act to $\mathcal{A}$.
– $\underline{\mathcal{O}_{\mathsf{Dec}}(\mathsf{act}):}$ If $\mathsf{act} \in Q$, $\mathcal{B}_1$ returns $\perp$. Otherwise, $\mathcal{B}_1$ parses $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$, $\overline{\mathsf{sk} = (\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})}$, and sets $x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$. If $\mathsf{ANIZK.Verify}(x, \pi) = 0$, $\mathcal{B}_1$ return $\perp$. Else, $\mathcal{B}_1$ returns $\mathsf{PKE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1)$ to $\mathcal{A}$.

Finally, $\mathcal{B}_1$ outputs $\mathcal{A}$'s final output b. That's the construction of $\mathcal{B}_1$.

$\mathcal{B}_1$'s simulation is Game 0 when the proving oracle is connected to $\mathcal{O}_{\mathsf{Real}}$ and Game 1 when the proving oracle is connected to $\mathcal{O}_{\mathsf{Sim}}$. Therefore,

$$|\Pr[E_1] - \Pr[E_0]| \leq \mathbf{Adv}^{\mathrm{razk}}_{\mathsf{ANIZK},\mathcal{B}_1,\mathcal{S}_{\mathsf{ana}}}(\lambda). \tag{18}$$

Game 2. Game 2 is the same as Game 1, except that its decryption oracle uses $\mathsf{PKE.Dec}(\mathsf{sk}_2, \mathsf{ct}_2)$. Let Bad_1 be the event that $\mathcal{A}$ submits a ciphertext $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi) \notin Q$ to decryption oracle $\mathcal{O}_{\mathsf{Dec}}$ such that, for $x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$,

$$\mathsf{ANIZK.Verify}(x, \pi) = 1, \mathsf{PKE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1) \neq \mathsf{PKE.Dec}(\mathsf{sk}_2, \mathsf{ct}_2).$$

Game 1 and Game 2 are identical until this Bad_1 occurs. Consequently,

$$|\Pr[E_2] - \Pr[E_1]| \leq \Pr[\mathsf{Bad}_1]. \tag{19}$$

We bound $\Pr[\mathsf{Bad}_1]$ by constructing an adversary $\mathcal{B}_2$ for the recipient anamorphic simulation soundness game, as defined in Definition 24. On receiving $(\mathsf{crs}, \mathsf{dk})$ from its challenger, $\mathcal{B}_2$ generates both PKE key pairs $(\mathsf{pk}_1, \mathsf{sk}_1)$ and $(\mathsf{pk}_2, \mathsf{sk}_2)$, and gives $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$ and dk to $\mathcal{A}$. It initializes the challenge set $Q := \emptyset$ and answers the oracles as follows.

– $\underline{\mathcal{O}_{\mathsf{chal}}(\mathsf{m}_0, \mathsf{m}_1, \mathsf{am}):}$ If $|\mathsf{m}_0| \neq |\mathsf{m}_1|$ or $\mathsf{am} \notin \mathcal{AM}$, $\mathcal{B}_2$ returns $\perp$. Otherwise, it chooses two independent random tapes $(r_1, r_2) \leftarrow \mathcal{RS}^2_{\mathsf{PKE.Enc}}$, computes $\mathsf{ct}_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_1, \mathsf{m}_0; r_1)$ and $\mathsf{ct}_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, \mathsf{m}_0; r_2)$. Then $\mathcal{B}_2$ sends $(x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2), \mathsf{am})$ to its own simulation oracle in the recipient anamorphic simulation soundness game. Assume π is the response of the simulation oracle. $\mathcal{B}_2$ sets $\mathsf{act} := (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ and $Q = Q \cup \{\mathsf{act}\}$. Finally, $\mathcal{B}_2$ returns act to $\mathcal{A}$.
– $\underline{\mathcal{O}_{\mathsf{Dec}}(\mathsf{act}):}$ If $\mathsf{act} \in Q$, $\mathcal{B}_2$ returns $\perp$. Otherwise, $\mathcal{B}_2$ parses $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$, $\overline{\mathsf{sk} = (\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})}$, and sets $x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$. If $\mathsf{ANIZK.Verify}(x, \pi) = 0$, $\mathcal{B}_2$ return $\perp$. Else, $\mathcal{B}_2$ computes $d_1 = \mathsf{PKE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1)$ and $d_2 = \mathsf{PKE.Dec}(\mathsf{sk}_2, \mathsf{ct}_2)$. If $d_1 \neq d_2$, it terminates and outputs $(x^*, \pi^*) = (x, \pi)$ to its challenger. If $d_1 = d_2$, it returns d_1 to $\mathcal{A}$.

If no such query (a query of $\mathcal{O}_{\mathsf{Dec}}$ such that $\mathcal{B}_2$ outputs (x, π)) occurs, $\mathcal{B}_2$ outputs a random pair (x^*, π^*).

When Bad_1 occurs, let $\mathsf{act}^* = (\mathsf{ct}_1^*, \mathsf{ct}_2^*, \pi^*)$ be the decryption oracle query of $\mathcal{A}$. $\mathcal{B}_2$ outputs $(x^* = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1^*, \mathsf{ct}_2^*), \pi^*)$ as its final output to its challenger. The perfect correctness of PKE implies $x \notin \mathcal{L}_{\mathcal{R}_{\mathsf{NY}}}$. Moreover, the fixed public keys and the complete ciphertext representation give the correspondence

$$(\mathsf{ct}_1, \mathsf{ct}_2, \pi) \in Q \quad \Longleftrightarrow \quad ((\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2), \pi) \in Q_{\mathsf{Sim}},$$

where Q_{Sim} is the set maintained by the RASS challenger. Thus, the output satisfies $\mathsf{Verify}(x^*, \pi^*) = 1$, $x^* \notin \mathcal{L}_{\mathcal{R}_{\mathsf{NY}}}$, and $(x^*, \pi^*) \notin Q_{\mathsf{Sim}}$. It follows that

$$\Pr[\mathsf{Bad}_1] \leq \mathbf{Adv}^{\mathrm{rass}}_{\mathsf{ANIZK}, \mathcal{B}_2, \mathcal{S}_{\mathsf{ana}}}(\lambda). \tag{20}$$

Game 3. Game 3 is the same as Game 2, except that the first PKE component of every challenge now encrypts m_1; the second PKE component still encrypts m_0.

To bound the hop, we construct a PPT adversary $\mathcal{B}_3$, who attempts to break the IND-CPA security of a PKE scheme.

After $\mathcal{B}_3$ receives pk_1 from its challenger, it runs $(\mathsf{crs}, \tau) \leftarrow \mathcal{S}_{\mathsf{ana}}.\mathsf{Setup}(1^\lambda)$, $\mathsf{dk} \leftarrow \mathsf{ANIZK.aKeyGen}(1^\lambda)$, $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, sets $\mathsf{pk} := (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$, and sends $(\mathsf{pk}, \mathsf{dk})$ to $\mathcal{A}$. When $\mathcal{A}$ queries the oracles $\mathcal{O}_{\mathsf{chal}}$ or $\mathcal{O}_{\mathsf{Dec}}$, $\mathcal{B}_3$ answers $\mathcal{A}$ as follows:

- $\underline{\mathcal{O}_{\mathsf{chal}}(\mathsf{m}_0, \mathsf{m}_1, \mathsf{am}) :}$ If $|\mathsf{m}_0| \neq |\mathsf{m}_1|$ or $\mathsf{am} \notin \mathcal{AM}$, $\mathcal{B}_3$ returns $\perp$. Otherwise, it chooses a random $r_2 \leftarrow \mathcal{RS}_{\mathsf{PKE.Enc}}$. $\mathcal{B}_3$ sends $(\mathsf{m}_0, \mathsf{m}_1)$ to its own public key encryption oracle. We assume that ct_1 is the response of the challenger in the IND-CPA game. Then $\mathcal{B}_3$ computes $\mathsf{ct}_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, \mathsf{m}_0; r_2)$. Next, $\mathcal{B}_3$ runs $\mathcal{S}_{\mathsf{ana}}.\mathsf{aProve}(\tau, x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2), \mathsf{dk}, \mathsf{am})$ to obtain a proof π. $\mathcal{B}_3$ sets $\mathsf{act} := (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ and $Q = Q \cup \{\mathsf{act}\}$. Finally, $\mathcal{B}_3$ returns act to $\mathcal{A}$.
- $\underline{\mathcal{O}_{\mathsf{Dec}}(\mathsf{act}) :}$ If $\mathsf{act} \in Q$, $\mathcal{B}_3$ returns $\perp$. Otherwise, $\mathcal{B}_3$ parses $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$, and sets $x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$. If $\mathsf{ANIZK.Verify}(x, \pi) = 0$, $\mathcal{B}_3$ return $\perp$. Else, $\mathcal{B}_3$ returns $\mathsf{PKE.Dec}(\mathsf{sk}_2, \mathsf{ct}_2)$ to $\mathcal{A}$.

If the challenger in the game of IND-CPA security of PKE chooses $b_{\$} = 0$, then $\mathcal{A}$'s view is the same as in Game 2. If the challenger chooses $b_{\$} = 1$, then $\mathcal{A}$'s view is the same as in Game 3. Therefore,

$$|\Pr[E_3] - \Pr[E_2]| \leq 2 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE}, \mathcal{B}_3}(\lambda). \tag{21}$$

Game 4. Game 4 is the same as Game 3, except that its decryption oracle uses $\underline{\mathsf{PKE.Dec}}(\mathsf{sk}_1, \mathsf{ct}_1)$. Let Bad_2 be the event that $\mathcal{A}$ submits a ciphertext $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi) \notin Q$ to decryption oracle $\mathcal{O}_{\mathsf{Dec}}$ such that, for $x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$,

$$\mathsf{ANIZK.Verify}(x, \pi) = 1, \mathsf{PKE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1) \neq \mathsf{PKE.Dec}(\mathsf{sk}_2, \mathsf{ct}_2).$$

Game 4 and Game 3 are identical until this Bad_2 occurs. Consequently,

$$|\Pr[E_4] - \Pr[E_3]| \leq \Pr[\mathsf{Bad}_2]. \tag{22}$$

Similar to the reduction between Game 1 and Game 2, we can construct a PPT adversary $\mathcal{B}_4$ for the recipient anamorphic simulation soundness game to bound the hop. Therefore,

$$|\Pr[E_4] - \Pr[E_3]| \leq \Pr[\mathsf{Bad}_2] \leq \mathbf{Adv}^{\mathrm{rass}}_{\mathsf{ANIZK},\mathcal{B}_4,\mathcal{S}_{\mathsf{ana}}}(\lambda). \tag{23}$$

Game 5. Game 5 is the same as Game 4, except that the second PKE component of every challenge is also changed to an encryption of m_1.

Similar to the reduction between Game 2 and Game 3, we can construct a PPT adversary $\mathcal{B}_5$ for the IND-CPA game of a PKE scheme to bound the hop. Therefore,

$$|\Pr[E_5] - \Pr[E_4]| \leq 2 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{B}_5}(\lambda). \tag{24}$$

Game 6. Game 6 replaces the simulated crs and simulated challenge proofs in Game 5 with an honest crs and honest anamorphic proofs.

To bound this hop, we construct a PPT adversary $\mathcal{B}_6$ in the recipient anamorphic zero knowledge game in Definition 13.

After $\mathcal{B}_6$ receives the $(\mathsf{crs}, \mathsf{dk})$ from its challenger, it generates two PKE pairs $(\mathsf{pk}_1, \mathsf{sk}_1)$, $(\mathsf{pk}_2, \mathsf{sk}_2)$, sets $\mathsf{pk} := (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$, $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$, and sends $(\mathsf{pk}, \mathsf{dk})$ to $\mathcal{A}$. When $\mathcal{A}$ queries the oracles $\mathcal{O}_{\mathsf{chal}}$ or $\mathcal{O}_{\mathsf{Dec}}$, $\mathcal{B}_6$ answers $\mathcal{A}$ as follows:

- $\mathcal{O}_{\mathsf{chal}}(\mathsf{m}_0, \mathsf{m}_1, \mathsf{am})$: If $|\mathsf{m}_0| \neq |\mathsf{m}_1|$ or $\mathsf{am} \notin \mathcal{AM}$, $\mathcal{B}_6$ returns $\perp$. Otherwise, it chooses two independent random tapes $(r_1, r_2) \leftarrow \mathcal{RS}^2_{\mathsf{PKE.Enc}}$, computes $\mathsf{ct}_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_1, \mathsf{m}_1; r_1)$ and $\mathsf{ct}_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, \mathsf{m}_1; r_2)$. Then $\mathcal{B}_6$ sends $(x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2), w = (\mathsf{m}_1, r_1, r_2), \mathsf{am})$ to its own proving oracle in the recipient anamorphic zero-knowledge game. Assume π is the response of the proving oracle. $\mathcal{B}_6$ sets $\mathsf{act} := (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ and $Q = Q \cup \{\mathsf{act}\}$. Finally, $\mathcal{B}_6$ returns act to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{Dec}}(\mathsf{act})$: If $\mathsf{act} \in Q$, $\mathcal{B}_6$ returns $\perp$. Otherwise, $\mathcal{B}_6$ parses $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$, $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$, and sets $x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$. If $\mathsf{ANIZK.Verify}(x, \pi) = 0$, $\mathcal{B}_6$ return $\perp$. Else, $\mathcal{B}_6$ returns $\mathsf{PKE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1)$ to $\mathcal{A}$.

Finally, $\mathcal{B}_6$ outputs $\mathcal{A}$'s final output b. $\mathcal{B}_6$'s simulation is Game 5 when the proving oracle is connected to $\mathcal{O}_{\mathsf{Sim}}$ and Game 6 when the proving oracle is connected to $\mathcal{O}_{\mathsf{Real}}$.

Therefore,

$$|\Pr[E_6] - \Pr[E_5]| \leq \mathbf{Adv}^{\mathrm{razk}}_{\mathsf{ANIZK},\mathcal{B}_6,\mathcal{S}_{\mathsf{ana}}}(\lambda). \tag{25}$$

Game 6 is exactly $\mathsf{G}^{\text{SIND-aCCA1}}_{\mathsf{RAME},\mathcal{A},1}(\lambda)$, and hence

$$\Pr[E_6] = \Pr[\mathsf{G}^{\text{SIND-aCCA1}}_{\mathsf{RAME},\mathcal{A},1}(\lambda) = 1]. \tag{26}$$

Finally, the triangle inequality, together with Eqs. (18)–(26), yields

$$\begin{aligned}\mathbf{Adv}^{\text{SIND-aCCA1}}_{\mathsf{RAME},\mathcal{A}}(\lambda) \leq\ & \mathbf{Adv}^{\mathrm{razk}}_{\mathsf{ANIZK},\mathcal{B}_1,\mathcal{S}_{\mathsf{ana}}}(\lambda) + \mathbf{Adv}^{\mathrm{rass}}_{\mathsf{ANIZK},\mathcal{B}_2,\mathcal{S}_{\mathsf{ana}}}(\lambda) + 2 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{B}_3}(\lambda) \\ & + \mathbf{Adv}^{\mathrm{rass}}_{\mathsf{ANIZK},\mathcal{B}_4,\mathcal{S}_{\mathsf{ana}}}(\lambda) + 2 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{B}_5}(\lambda) + \mathbf{Adv}^{\mathrm{razk}}_{\mathsf{ANIZK},\mathcal{B}_6,\mathcal{S}_{\mathsf{ana}}}(\lambda) \\ & \leq \mathsf{negl}(\lambda)\end{aligned}$$

Thus, RAME satisfies SIND-aCCA1 security.

E.3 Proof of SIND-nCCA

Proof. Let $\mathcal{S} = (\mathcal{S}.\mathsf{Setup}, \mathcal{S}.\mathsf{Prove})$ be the simulator in the zero knowledge security of ANIZK. For $i \in \{0,1,2,3,4,5,6\}$, let E_i denote the event that the output of Game i is 1.

Game 0. This is the original game $\mathsf{G}^{\text{SIND-nCCA}}_{\mathsf{RAME},\mathcal{A},0}(\lambda)$. Thus, both PKE components of each challenge encrypt m_0, and the attached proof is an honest proof. Hence,

$$\Pr[E_0] = \Pr[\mathsf{G}^{\text{SIND-nCCA}}_{\mathsf{RAME},\mathcal{A},0}(\lambda) = 1]. \tag{27}$$

Game 1. Game 1 is the same as Game 0, except that the challenger uses $(\mathsf{crs}, \tau) \leftarrow \mathcal{S}.\mathsf{Setup}(1^\lambda)$ and replaces every honest challenge proof by a proof generated with $\mathcal{S}.\mathsf{Prove}(\tau, x)$.

To bound this hop, we construct a PPT adversary $\mathcal{B}_1$ in the zero knowledge game in Definition 4.

After $\mathcal{B}_1$ receives the crs from its challenger, it generates a double key $\mathsf{dk} \leftarrow \mathsf{ANIZK.aKeyGen}(1^\lambda)$ and two PKE pairs $(\mathsf{pk}_1, \mathsf{sk}_1)$, $(\mathsf{pk}_2, \mathsf{sk}_2)$, sets $\mathsf{pk} := (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$, $\mathsf{sk} := (\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$, and sends $(\mathsf{pk}, \mathsf{dk})$ to $\mathcal{A}$. When $\mathcal{A}$ queries the oracles $\mathcal{O}_{\mathsf{chal}}$ or $\mathcal{O}_{\mathsf{Dec}}$, $\mathcal{B}_1$ answers $\mathcal{A}$ as follows:

- $\mathcal{O}_{\mathsf{chal}}(\mathsf{m}_0, \mathsf{m}_1)$: If $|\mathsf{m}_0| \neq |\mathsf{m}_1|$, $\mathcal{B}_1$ returns $\perp$. Otherwise, it chooses two randomness $(r_1, r_2) \leftarrow \mathcal{RS}^2_{\mathsf{PKE.Enc}}$, computes $\mathsf{ct}_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_1, \mathsf{m}_0; r_1)$ and $\mathsf{ct}_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, \mathsf{m}_0; r_2)$. Then $\mathcal{B}_1$ sends $(x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2), w = (\mathsf{m}_0, r_1, r_2))$ to its own proving oracle in the zero knowledge game. Assume π is the response of the proving oracle. $\mathcal{B}_1$ sets $\mathsf{act} := (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ and $Q = Q \cup \{\mathsf{act}\}$. Finally, $\mathcal{B}_1$ returns act to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{Dec}}(\mathsf{act})$: If $\mathsf{act} \in Q$, $\mathcal{B}_1$ returns $\perp$. Otherwise, $\mathcal{B}_1$ parses $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$, $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$, and sets $x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$. If $\mathsf{ANIZK.Verify}(x, \pi) = 0$, $\mathcal{B}_1$ return $\perp$. Else, $\mathcal{B}_1$ returns $\mathsf{PKE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1)$ to $\mathcal{A}$.

Finally, $\mathcal{B}_1$ outputs $\mathcal{A}$'s final output b.

$\mathcal{B}_1$'s simulation is Game 0 when the proving oracle is connected to $\mathcal{O}_{\mathsf{Prove}}$ and Game 1 when the proving oracle is connected to $\mathcal{O}_{\mathsf{Sim}}$. Therefore,

$$|\Pr[E_1] - \Pr[E_0]| \leq \mathbf{Adv}^{\text{zk}}_{\mathsf{ANIZK},\mathcal{B}_1,\mathcal{S}}(\lambda). \tag{28}$$

Game 2. Game 2 is the same as Game 1, except that its decryption oracle uses $\mathsf{PKE.Dec}(\mathsf{sk}_2, \mathsf{ct}_2)$. Let Bad_1 be the event that $\mathcal{A}$ submits a ciphertext $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi) \notin Q$ to decryption oracle $\mathcal{O}_{\mathsf{Dec}}$ such that, for $x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$,

$$\mathsf{ANIZK.Verify}(x, \pi) = 1, \mathsf{PKE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1) \neq \mathsf{PKE.Dec}(\mathsf{sk}_2, \mathsf{ct}_2).$$

Game 1 and Game 2 are identical until this Bad_1 occurs. Consequently,

$$|\Pr[E_2] - \Pr[E_1]| \leq \Pr[\mathsf{Bad}_1] \tag{29}$$

We bound $\Pr[\mathsf{Bad}_1]$ by constructing an adversary $\mathcal{B}_2$ for the simulation soundness game for NIZK, as defined in Definition 18. On receiving crs from its challenger, $\mathcal{B}_2$ generates both PKE key pairs $(\mathsf{pk}_1, \mathsf{sk}_1)$ and $(\mathsf{pk}_2, \mathsf{sk}_2)$, $\mathsf{dk} \leftarrow \mathsf{ANIZK.aKeyGen}(1^\lambda)$ and gives $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$ and dk to $\mathcal{A}$. It initializes the challenge set $Q := \emptyset$ and answers the oracles as follows.

– $\underline{\mathcal{O}_{\mathsf{chal}}(\mathsf{m}_0, \mathsf{m}_1)}$: If $|\mathsf{m}_0| \neq |\mathsf{m}_1|$, $\mathcal{B}_2$ returns $\perp$. Otherwise, it chooses two independent random tapes $(r_1, r_2) \leftarrow \mathcal{RS}^2_{\mathsf{PKE.Enc}}$, computes $\mathsf{ct}_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_1, \mathsf{m}_0; r_1)$ and $\mathsf{ct}_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, \mathsf{m}_0; r_2)$. Then $\mathcal{B}_2$ sends $x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$ to its own simulation oracle in the simulation soundness game. Assume π is the response of the simulation oracle. $\mathcal{B}_2$ sets $\mathsf{act} := (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ and $Q = Q \cup \{\mathsf{act}\}$. Finally, $\mathcal{B}_2$ returns act to $\mathcal{A}$.
– $\underline{\mathcal{O}_{\mathsf{Dec}}(\mathsf{act})}$: If $\mathsf{act} \in Q$, $\mathcal{B}_2$ returns $\perp$. Otherwise, $\mathcal{B}_2$ parses $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$, $\overline{\mathsf{sk} = (\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})}$, and sets $x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$. If $\mathsf{ANIZK.Verify}(x, \pi) = 0$, $\mathcal{B}_2$ return $\perp$. Else, $\mathcal{B}_2$ computes $d_1 = \mathsf{PKE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1)$ and $d_2 = \mathsf{PKE.Dec}(\mathsf{sk}_2, \mathsf{ct}_2)$. If $d_1 \neq d_2$, it terminates and outputs (x, π) to its challenger. If $d_1 = d_2$, it returns d_1 to $\mathcal{A}$.

If no such query (a query of $\mathcal{O}_{\mathsf{Dec}}$ such that $\mathcal{B}_2$ outputs $(x^*, \pi^*) = (x, \pi)$) occurs, $\mathcal{B}_2$ outputs a random pair (x^*, π^*).

When Bad_1 occurs, let $\mathsf{act}^* = (\mathsf{ct}_1^*, \mathsf{ct}_2^*, \pi^*)$ as the decryption oracle query of $\mathcal{A}$. $\mathcal{B}_2$ outputs $(x^* = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1^*, \mathsf{ct}_2^*), \pi^*)$ as its final output to its challenger. The perfect correctness of PKE implies $x \notin \mathcal{L}_{\mathcal{R}_{\mathsf{NY}}}$. Moreover, the fixed public keys and the complete ciphertext representation give the correspondence

$$(\mathsf{ct}_1, \mathsf{ct}_2, \pi) \in Q \quad \Longleftrightarrow \quad ((\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2), \pi) \in Q_{\mathsf{Sim}},$$

where Q_{Sim} is the set maintained by the simulation soundness challenger. Thus, the output satisfies $\mathsf{Verify}(x^*, \pi^*) = 1$, $x^* \notin \mathcal{L}_{\mathcal{R}_{\mathsf{NY}}}$, and $(x^*, \pi^*) \notin Q_{\mathsf{Sim}}$. It follows that

$$\Pr[\mathsf{Bad}_1] \leq \mathbf{Adv}^{\mathrm{ss}}_{\mathsf{ANIZK}, \mathcal{B}_2, \mathcal{S}}(\lambda) \tag{30}$$

Game 3. Game 3 is the same as Game 2, except that the first PKE component of every challenge now encrypts m_1; the second PKE component still encrypts m_0.

To bound the hop, we construct a PPT adversary $\mathcal{B}_3$, who attempts to break the IND-CPA security of a PKE scheme.

After $\mathcal{B}_3$ receives pk_1 from its challenger, it runs $(\mathsf{crs}, \tau) \leftarrow \mathcal{S}.\mathsf{Setup}(1^\lambda)$, $\mathsf{dk} \leftarrow \mathsf{ANIZK.aKeyGen}(1^\lambda)$, $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, sets $\mathsf{pk} := (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$, and sends $(\mathsf{pk}, \mathsf{dk})$ to $\mathcal{A}$.

When $\mathcal{A}$ queries the oracles $\mathcal{O}_{\mathsf{chal}}$ or $\mathcal{O}_{\mathsf{Dec}}$, $\mathcal{B}_3$ answers $\mathcal{A}$ as follows:

– $\underline{\mathcal{O}_{\mathsf{chal}}(\mathsf{m}_0, \mathsf{m}_1)}$: If $|\mathsf{m}_0| \neq |\mathsf{m}_1|$, $\mathcal{B}_3$ returns $\perp$. Otherwise, it chooses a random $\overline{r_2 \leftarrow \mathcal{RS}_{\mathsf{PKE.Enc}}}$. $\mathcal{B}_3$ sends $(\mathsf{m}_0, \mathsf{m}_1)$ to its own public key encryption oracle. We assume that ct_1 is the response of the challenger in the game of the IND-CPA security of PKE. Then $\mathcal{B}_3$ computes $\mathsf{ct}_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, \mathsf{m}_0; r_2)$. Next, $\mathcal{B}_3$ runs $\mathcal{S}.\mathsf{Prove}(\tau, x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2))$ to obtain a proof π. $\mathcal{B}_3$ sets $\mathsf{act} := (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ and $Q = Q \cup \{\mathsf{act}\}$. Finally, $\mathcal{B}_3$ returns act to $\mathcal{A}$.
– $\underline{\mathcal{O}_{\mathsf{Dec}}(\mathsf{act})}$: If $\mathsf{act} \in Q$, $\mathcal{B}_3$ returns $\perp$. Otherwise, $\mathcal{B}_3$ parses $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$, and sets $x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$. If $\mathsf{ANIZK.Verify}(x, \pi) = 0$, $\mathcal{B}_3$ return $\perp$. Else, $\mathcal{B}_3$ returns $\mathsf{PKE.Dec}(\mathsf{sk}_2, \mathsf{ct}_2)$ to $\mathcal{A}$.

If the challenger in the game of IND-CPA security of PKE chooses $b_{\$} = 0$, then $\mathcal{A}$'s view is the same as in Game 2. If the challenger chooses $b_{\$} = 1$, then $\mathcal{A}$'s view is the same as in Game 3. Therefore,

$$|\Pr[E_3] - \Pr[E_2]| \leq 2 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{B}_3}(\lambda). \tag{31}$$

Game 4. Game 4 is the same as Game 3, except that its decryption oracle uses $\overline{\mathsf{PKE.Dec}}(\mathsf{sk}_1, \mathsf{ct}_1)$. Let Bad_2 be the event that $\mathcal{A}$ submits a ciphertext $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi) \notin Q$ to decryption oracle $\mathcal{O}_{\mathsf{Dec}}$ such that, for $x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$,

$$\mathsf{ANIZK.Verify}(x, \pi) = 1, \mathsf{PKE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1) \neq \mathsf{PKE.Dec}(\mathsf{sk}_2, \mathsf{ct}_2).$$

Game 4 and Game 3 are identical until this Bad_2 occurs. Consequently,

$$|\Pr[E_4] - \Pr[E_3]| \leq \Pr[\mathsf{Bad}_2]. \tag{32}$$

Similar to the reduction between Game 1 and Game 2, we can construct a PPT adversary $\mathcal{B}_4$ for the simulation soundness game to bound the hop. Therefore,

$$|\Pr[E_4] - \Pr[E_3]| \leq \Pr[\mathsf{Bad}_2] \leq \mathbf{Adv}^{\text{ss}}_{\mathsf{ANIZK},\mathcal{B}_4,\mathcal{S}}(\lambda). \tag{33}$$

Game 5. Game 5 is the same as Game 4, except that the second PKE component of every challenge is also changed to an encryption of m_1.

Similar to the reduction between Game 2 and Game 3, we can construct a PPT adversary $\mathcal{B}_5$ for the IND-CPA game of a PKE scheme to bound the hop. Therefore,

$$|\Pr[E_5] - \Pr[E_4]| \leq 2 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{B}_5}(\lambda). \tag{34}$$

Game 6. Game 6 replaces the simulated crs and simulated challenge proofs in Game 5 with an honest crs and honest proofs.

To bound this hop, we construct a PPT adversary $\mathcal{B}_6$ in the zero knowledge game in Definition 4.

After $\mathcal{B}_6$ receives the crs from its challenger, it generates a double key $\mathsf{dk} \leftarrow \mathsf{ANIZK.aKeyGen}(1^\lambda)$, generates two PKE pairs $(\mathsf{pk}_1, \mathsf{sk}_1)$, $(\mathsf{pk}_2, \mathsf{sk}_2)$, sets $\mathsf{pk} := (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$, $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$, and sends $(\mathsf{pk}, \mathsf{dk})$ to $\mathcal{A}$. When $\mathcal{A}$ queries the oracles $\mathcal{O}_{\mathsf{chal}}$ or $\mathcal{O}_{\mathsf{Dec}}$, $\mathcal{B}_6$ answers $\mathcal{A}$ as follows:

- $\underline{\mathcal{O}_{\mathsf{chal}}(\mathsf{m}_0, \mathsf{m}_1)}$: If $|\mathsf{m}_0| \neq |\mathsf{m}_1|$, $\mathcal{B}_6$ returns $\perp$. Otherwise, it chooses two randomness $(r_1, r_2) \leftarrow \mathcal{RS}^2_{\mathsf{PKE.Enc}}$, computes $\mathsf{ct}_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_1, \mathsf{m}_1; r_1)$ and $\mathsf{ct}_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, \mathsf{m}_1; r_2)$. Then $\mathcal{B}_6$ sends $(x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2), w = (\mathsf{m}_1, r_1, r_2))$ to its own proving oracle in the zero-knowledge game. Assume π is the response of the proving oracle. $\mathcal{B}_6$ sets $\mathsf{act} := (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ and $Q = Q \cup \{\mathsf{act}\}$. Finally, $\mathcal{B}_6$ returns act to $\mathcal{A}$.
- $\underline{\mathcal{O}_{\mathsf{Dec}}(\mathsf{act})}$: If $\mathsf{act} \in Q$, $\mathcal{B}_6$ returns $\perp$. Otherwise, $\mathcal{B}_6$ parses $\mathsf{act} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$, $\overline{\mathsf{sk} = (\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})}$, and sets $x = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$. If $\mathsf{ANIZK.Verify}(x, \pi) = 0$, $\mathcal{B}_6$ return $\perp$. Else, $\mathcal{B}_6$ returns $\mathsf{PKE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1)$ to $\mathcal{A}$.

$\underline{\mathsf{G}^{\mathrm{anam}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda)}$:
$b \leftarrow \{0,1\}$, $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{apk},\mathsf{ask}) \leftarrow \mathsf{aKeyGen}(1^\lambda)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{chal}}}(\mathsf{crs},\mathsf{apk})$
Return $(b = b')$

$\underline{\mathcal{O}_{\mathsf{chal}}(x,w,\mathsf{am})}$:
If $(x,w) \notin \mathcal{R}$ or $\mathsf{am} \notin \mathcal{AM}$: Return $\bot$
If $b = 0$:
  $r \leftarrow \mathcal{RS}_{\mathsf{Prove}}$, $\pi \leftarrow \mathsf{Prove}(x,w;r)$, $r^* \leftarrow r$
If $b = 1$:
  $r_{\mathsf{a}} \leftarrow \mathcal{RS}_{\mathsf{aProve}}$, $\pi \leftarrow \mathsf{aProve}(x,w,\mathsf{apk},\mathsf{am};r_{\mathsf{a}})$
  $r_{\mathsf{fake}} \leftarrow \mathsf{Equiv}(x,w,\mathsf{apk},\mathsf{am},r_{\mathsf{a}})$, $r^* \leftarrow r_{\mathsf{fake}}$
Return (π, r^*)

Fig. 16: Anamorphic security game for asymmetric anamorphic NIZK

Finally, $\mathcal{B}_6$ outputs $\mathcal{A}$'s final output b.

$\mathcal{B}_6$'s simulation is Game 5 when the proving oracle is connected to $\mathcal{O}_{\mathsf{Sim}}$ and Game 6 when the proving oracle is connected to $\mathcal{O}_{\mathsf{Real}}$. Therefore,

$$|\Pr[E_6] - \Pr[E_5]| \leq \mathbf{Adv}^{\mathrm{zk}}_{\mathsf{ANIZK},\mathcal{B}_6,\mathcal{S}}(\lambda). \tag{35}$$

Game 6 is exactly $\mathsf{G}^{\text{SIND-nCCA}}_{\mathsf{RAME},\mathcal{A},1}(\lambda)$, and hence

$$\Pr[E_6] = \Pr[\mathsf{G}^{\text{SIND-nCCA}}_{\mathsf{RAME},\mathcal{A},1}(\lambda) = 1]. \tag{36}$$

Finally, the triangle inequality, together with Eqs. (28)–(36), yields

$$\begin{aligned}\mathbf{Adv}^{\text{SIND-nCCA}}_{\mathsf{RAME},\mathcal{A}}(\lambda) &\leq \mathbf{Adv}^{\mathrm{zk}}_{\mathsf{ANIZK},\mathcal{B}_1,\mathcal{S}}(\lambda) + \mathbf{Adv}^{\mathrm{ss}}_{\mathsf{ANIZK},\mathcal{B}_2,\mathcal{S}}(\lambda) + 2\cdot\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{B}_3}(\lambda)\\ &\quad + \mathbf{Adv}^{\mathrm{ss}}_{\mathsf{ANIZK},\mathcal{B}_4,\mathcal{S}}(\lambda) + 2\cdot\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{B}_5}(\lambda) + \mathbf{Adv}^{\mathrm{zk}}_{\mathsf{ANIZK},\mathcal{B}_6,\mathcal{S}}(\lambda)\\ &\leq \mathsf{negl}(\lambda)\end{aligned}$$

Thus, RAME satisfies SIND-nCCA security.

F Asymmetric Anamorphic NIZK

F.1 Security notions

Definition 28 (Anamorphic Security). *An ASY-ANIZK scheme* ASY-ANIZK *is anamorphically secure, if there exists a PPT algorithm* Equiv *(which takes* $(x,w,\mathsf{apk},\mathsf{am},r_{\mathsf{a}} \in \mathcal{RS}_{\mathsf{aProve}})$ *as input and outputs* $r_{\mathsf{fake}} \in \mathcal{RS}_{\mathsf{Prove}}$*) such that for any PPT adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\mathrm{anam}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda) = |\Pr[\mathsf{G}^{\mathrm{anam}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\mathrm{anam}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda)$ *are defined in Fig. 16.*

Definition 29 (Recipient Anamorphic Zero Knowledge). *An ASY-ANIZK scheme* ASY-ANIZK *is recipient anamorphic zero-knowledge, if there exists a PPT simulator* $\mathcal{S}_{\mathsf{ana}} = (\mathcal{S}_{\mathsf{ana}}.\mathsf{Setup}, \mathcal{S}_{\mathsf{ana}}.\mathsf{aProve})$ *such that for any PPT adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\mathrm{razk}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda) = |\Pr[\mathsf{G}^{\mathrm{razk}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{G}^{\mathrm{razk}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\mathrm{razk}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda)$ *and* $\mathsf{G}^{\mathrm{razk}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda)$ *are defined in Fig. 17.*

$\underline{\mathsf{G}^{\mathrm{razk}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda)}$:
$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{aKeyGen}(1^\lambda)$
Return $b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Real}}}(\mathsf{crs}, \mathsf{apk}, \mathsf{ask})$

$\underline{\mathcal{O}_{\mathsf{Real}}(x, w, \mathsf{am})}$:
If $(x, w) \notin \mathcal{R}$ or $\mathsf{am} \notin \mathcal{AM}$: Return $\bot$
Return $\mathsf{aProve}(x, w, \mathsf{apk}, \mathsf{am})$

$\underline{\mathsf{G}^{\mathrm{razk}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda)}$:
$(\mathsf{crs}, \tau) \leftarrow \mathcal{S}_{\mathsf{ana}}.\mathsf{Setup}(1^\lambda)$, $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{aKeyGen}(1^\lambda)$
Return $b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sim}}}(\mathsf{crs}, \mathsf{apk}, \mathsf{ask})$

$\underline{\mathcal{O}_{\mathsf{Sim}}(x, w, \mathsf{am})}$:
If $(x, w) \notin \mathcal{R}$ or $\mathsf{am} \notin \mathcal{AM}$: Return $\bot$
Return $\mathcal{S}_{\mathsf{ana}}.\mathsf{aProve}(\tau, x, \mathsf{apk}, \mathsf{am})$

Fig. 17: Recipient anamorphic zero knowledge game for asymmetric anamorphic NIZK

$\underline{\mathsf{G}^{\mathrm{rase}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda)}$:
$Q := \emptyset$, $(\mathsf{crs}, \tau) \leftarrow \mathcal{S}_{\mathsf{ana}}.\mathsf{Setup}(1^\lambda)$, $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{aKeyGen}(1^\lambda)$
$(x^*, \pi^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sim}}}(\mathsf{crs}, \mathsf{apk}, \mathsf{ask})$, $w^* \leftarrow \mathsf{Ext}(\tau, \mathsf{apk}, \mathsf{ask}, x^*, \pi^*)$
If $(\mathsf{Verify}(x^*, \pi^*) \wedge ((x^*, \pi^*) \notin Q) \wedge ((x^*, w^*) \notin \mathcal{R}))$: Return 1
Return 0

$\underline{\mathcal{O}_{\mathsf{Sim}}(x, \mathsf{am})}$:
If $\mathsf{am} \notin \mathcal{AM}$: Return $\bot$
$\pi \leftarrow \mathcal{S}_{\mathsf{ana}}.\mathsf{aProve}(\tau, x, \mathsf{apk}, \mathsf{am})$, $Q = Q \cup \{(x, \pi)\}$
Return π

Fig. 18: Recipient anamorphic simulation extractability game for asymmetric anamorphic NIZK

Definition 30 (Recipient Anamorphic Simulation Extractability). *An ASY-ANIZK scheme* $\mathsf{ASY\text{-}ANIZK}$*, with a PPT simulator* $\mathcal{S}_{\mathsf{ana}} = (\mathcal{S}_{\mathsf{ana}}.\mathsf{Setup}, \mathcal{S}_{\mathsf{ana}}.\mathsf{aProve})$ *that satisfies Definition 29, is recipient anamorphic simulation extractable, if for any PPT adversary* $\mathcal{A}$*, there exists a PPT extractor* Ext *such that*

$$\mathbf{Adv}^{\mathrm{rase}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda) = \Pr[\mathsf{G}^{\mathrm{rase}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\mathrm{rase}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda)$ *is defined in Fig. 18.*

G Labeled NIZK

We employ a variant of NIZK that takes an additional public label (or context string). Binding a proof to specific associated data via labels is a well-established technique to achieve non-malleability and simulation extractability (e.g., as formalized in [4]).

Definition 31 (Labeled NIZK). *A labeled non-interactive zero knowledge proof system* LNIZK *for a relation* $\mathcal{R}$ *with label space* $\mathcal{LS}$ *is a tuple of PPT algorithms* $\mathsf{LNIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$*:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$*: The setup algorithm takes the security parameter* λ *as input, and then outputs a common reference string* crs*. Besides,* crs *is an implicit input on the following algorithms.*
- $\mathsf{Prove}(x, w, l; r) \to \pi'$*: The proving algorithm takes* $(x, w) \in \mathcal{R}$*, a public label* $l \in \mathcal{LS}$*, and randomness* $r \in \mathcal{RS}_{\mathsf{LNIZK}}$ *as input, and then outputs a proof* π'*.*
- $\mathsf{Verify}(x, \pi', l) \to \{0, 1\}$*: The verifying algorithm takes statement* x*, proof* π'*, and label* l *as input, and then outputs* 1 *(accept) or* 0 *(reject).*

We require the following properties.
Completeness. For all $(x, w) \in \mathcal{R}$, $l \in \mathcal{LS}$, $r \in \mathcal{RS}_{\mathsf{LNIZK}}$,

$$\Pr[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \pi \leftarrow \mathsf{Prove}(x, w, l; r) : \mathsf{Verify}(x, \pi) = 1] \geq 1 - \mathsf{negl}(\lambda).$$

Computational Soundness. For any PPT $\mathcal{A}$*, it holds that*

$$\Pr[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), (x, \pi, l) \leftarrow \mathcal{A}(\mathsf{crs}) : \mathsf{Verify}(x, \pi, l) = 1 \wedge x \notin \mathcal{L}_{\mathcal{R}}] \leq \mathsf{negl}(\lambda).$$

Labeled Zero Knowledge. There exists a PPT simulator $\mathcal{S} = (\mathcal{S}.\mathsf{Setup}, \mathcal{S}.\mathsf{Prove})$*, such that for any PPT adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\mathrm{lzk}}_{\mathsf{LNIZK},\mathcal{A},\mathcal{S}}(\lambda) = |\Pr[\mathsf{G}^{\mathrm{lzk}}_{\mathsf{LNIZK},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{G}^{\mathrm{lzk}}_{\mathsf{LNIZK},\mathcal{A},\mathcal{S}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\mathrm{lzk}}_{\mathsf{LNIZK},\mathcal{A}}(\lambda)$ *and* $\mathsf{G}^{\mathrm{lzk}}_{\mathsf{LNIZK},\mathcal{A},\mathcal{S}}(\lambda)$ *are defined in Fig. 19*

Then, we recall simulation extractability as formally defined below.

Definition 32 (Simulation Extractability). *A labeled NIZK scheme* LNIZK*, with a simulator* $\mathcal{S} = (\mathcal{S}.\mathsf{Setup}, \mathcal{S}.\mathsf{Prove})$ *satisfies Definition 31, achieves simulation extractability (SE) if for any PPT adversary* $\mathcal{A}$*, there exists a PPT extractor* Ext *such that the advantage:*

$$\mathbf{Adv}^{\mathrm{se}}_{\mathsf{LNIZK},\mathcal{A},\mathcal{S}}(\lambda) = \Pr[\mathsf{G}^{\mathrm{se}}_{\mathsf{LNIZK},\mathcal{A},\mathcal{S}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{G}^{\mathrm{se}}_{\mathsf{LNIZK},\mathcal{A},\mathcal{S}}(\lambda)$ *is defined in Fig. 19.*

$\underline{\mathsf{G}^{\mathrm{lzk}}_{\mathsf{LNIZK},\mathcal{A}}(\lambda)}$:	$\underline{\mathsf{G}^{\mathrm{lzk}}_{\mathsf{LNIZK},\mathcal{A},\mathcal{S}}(\lambda)}$:
$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$	$(\mathsf{crs}, \tau) \leftarrow \mathcal{S}.\mathsf{Setup}(1^\lambda)$
Return $b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Real}}}(\mathsf{crs})$	Return $b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sim}}}(\mathsf{crs})$
$\underline{\mathcal{O}_{\mathsf{Real}}(x, w, l)}$:	$\underline{\mathcal{O}_{\mathsf{Sim}}(x, w, l)}$:
If $(x, w) \notin \mathcal{R}$ or $l \notin \mathcal{LS}$: Return $\perp$	If $(x, w) \notin \mathcal{R}$ or $l \notin \mathcal{LS}$: Return $\perp$
Return $\mathsf{Prove}(x, w, l)$	Return $\mathcal{S}.\mathsf{Prove}(\tau, x, l)$

$\underline{\mathsf{G}^{\mathrm{se}}_{\mathsf{LNIZK},\mathcal{A},\mathcal{S}}(\lambda)}$:
$Q := \emptyset$, $(\mathsf{crs}, \tau) \leftarrow \mathcal{S}.\mathsf{Setup}(1^\lambda)$
$(x^*, l^*, \pi'^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sim}}}(\mathsf{crs})$, $w^* \leftarrow \mathsf{Ext}^{\mathcal{A}}(\tau, x^*, l^*, \pi'^*)$
If $(\mathsf{Verify}(x^*, \pi'^*, l^*) = 1 \wedge ((x^*, l^*, \pi'^*) \notin Q) \wedge ((x^*, w^*) \notin \mathcal{R}))$: Return 1
Return 0

$\underline{\mathcal{O}_{\mathsf{Sim}}(x, l)}$:
$\pi' \leftarrow \mathcal{S}.\mathsf{Prove}(\tau, x, l)$, $Q = Q \cup \{(x, l, \pi')\}$
Return π'

Fig. 19: Labeled zero knowledge and simulation extractability game for Labeled NIZK

H The proof of Theorem 3

Before proving the security of ASY-ANIZK, we note that the algorithms (Setup, Prove, Verify) of ASY-ANIZK in Fig. 9 constitute a standard NIZK proof system. This is immediate, as these three algorithms are identical to those of LNIZK, which already satisfies completeness, zero knowledge, and computational soundness as in Definition 4.

Now, we prove the four claimed properties (anamorphic security, weak robustness, recipient anamorphic zero knowledge, recipient anamorphic simulation extractability) separately. For your interest, the corresponding games are defined in Appendix F.1, and the construction is in Fig. 9.

H.1 Proof of Anamorphic Security

$\underline{\mathsf{Setup}(1^\lambda)}$:
$\mathsf{crs}_{\mathsf{LNIZK}} \leftarrow \mathsf{Setup}_{\mathsf{LNIZK}}(1^\lambda)$
Return $\mathsf{crs} := \mathsf{crs}_{\mathsf{LNIZK}}$

$\underline{\mathsf{Prove}(x, w; r_{\mathsf{std}})}$:
Parse $r_{\mathsf{std}} = (r, \rho)$
// where $r \leftarrow \mathcal{RS}_{\mathsf{LNIZK}}$ and $\rho \leftarrow \mathcal{LS}$
$\pi' \leftarrow \mathsf{LNIZK.Prove}(x, w, l = \rho; r)$
Return $\pi := (\pi', \rho)$

$\underline{\mathsf{Verify}(x, \pi)}$:
Parse $\pi = (\pi', \rho)$
Return $\mathsf{LNIZK.Verify}(x, \pi', l = \rho)$

$\underline{\mathsf{Equiv}(x, w, \mathsf{apk}, \mathsf{am}, r_{\mathsf{a}})}$:
Parse $r_{\mathsf{a}} = (r, s)$
$\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{apk}, \mathsf{am}; s)$
Return $r_{\mathsf{fake}} = (r, \mathsf{ct})$

$\underline{\mathsf{aKeyGen}(1^\lambda)}$:
$(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$
Return $\mathsf{apk}, \mathsf{ask}$

$\underline{\mathsf{aProve}(x, w, \mathsf{apk}, \mathsf{am}; r_{\mathsf{a}})}$:
Parse $r_{\mathsf{a}} = (r, s)$
// where $r \leftarrow \mathcal{RS}_{\mathsf{LNIZK}}$ and $s \leftarrow \mathcal{RS}_{\mathsf{PKE.Enc}}$
$\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{apk}, \mathsf{am}; s)$
$\pi' \leftarrow \mathsf{LNIZK.Prove}(x, w, l = \mathsf{ct}; r)$
Return $\pi^* := (\pi', \mathsf{ct})$

$\underline{\mathsf{aExtract}(x, \pi^*, \mathsf{ask})}$:
Parse $\pi^* = (\pi', \mathsf{ct})$
Return $\mathsf{PKE.Dec}(\mathsf{ask}, \mathsf{ct})$

Fig. 20: The generic construction of ASY-ANIZK with Equiv algorithm

Proof. We construct the Equiv algorithm in Fig. 20 and show its relation with other algorithms.

We first specify the equivocation algorithm. Write

$$\mathcal{RS}_{\mathsf{Prove}} = \mathcal{RS}_{\mathsf{LNIZK}} \times \mathcal{LS}, \qquad \mathcal{RS}_{\mathsf{aProve}} = \mathcal{RS}_{\mathsf{LNIZK}} \times \mathcal{RS}_{\mathsf{PKE.Enc}}.$$

For $r_{\mathsf{a}} = (r, s)$, define

$$\mathsf{Equiv}(x, w, \mathsf{apk}, \mathsf{am}, r_{\mathsf{a}}): \quad \mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{apk}, \mathsf{am}; s); \quad \text{return } r_{\mathsf{fake}} = (r, \mathsf{ct}).$$

For every valid (x, w), message am, and tape $r_{\mathsf{a}} = (r, s)$, aProve (running with inner randomness r_{a}) and Prove (running with inner randomness r_{fake}) both

utilize the same label $\mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{apk}, \mathsf{am}; s)$ and the same LNIZK randomness r. Thus,

$$\mathsf{Prove}\big(x, w; \mathsf{Equiv}(x, w, \mathsf{apk}, \mathsf{am}, r_{\mathsf{a}})\big) = \mathsf{aProve}(x, w, \mathsf{apk}, \mathsf{am}; r_{\mathsf{a}}). \tag{37}$$

Let $\mathcal{A}$ be a PPT adversary in the anamorphic security game. For $i \in \{0, 1\}$, let E_i be the event that $\mathcal{A}$ outputs 1 in Game i.
Game 0. The challenger generates $\mathsf{crs}_{\mathsf{LNIZK}} \leftarrow \mathsf{Setup}_{\mathsf{LNIZK}}(1^\lambda)$ and $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, then sends $(\mathsf{crs}, \mathsf{apk})$ to $\mathcal{A}$. The challenger answers $\mathcal{A}$'s query as follows.

- $\mathcal{O}_{\mathsf{chal}}(x, w, \mathsf{am})$: If $(x, w) \notin \mathcal{R}$ or $\mathsf{am} \notin \mathcal{AM}$, the game returns $\perp$. Otherwise, the challenger samples $r \leftarrow \mathcal{RS}_{\mathsf{LNIZK}}$ and $\rho \leftarrow \mathcal{LS}$, sets $r^* = (r, \rho)$, computes $\pi' \leftarrow \mathsf{LNIZK.Prove}(x, w, \rho; r)$, and sends $(\pi := (\pi', \rho), r^*)$ to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs a bit b'. This is the $b = 0$ branch of $\mathsf{G}^{\mathrm{anam}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda)$.
Game 1. Game 1 is similar to Game 0, except that the challenger answers the $\mathcal{A}$ query as follows.

- $\mathcal{O}_{\mathsf{chal}}(x, w, \mathsf{am})$: If $(x, w) \notin \mathcal{R}$ or $\mathsf{am} \notin \mathcal{AM}$, the game returns $\perp$. Otherwise, the challenger samples $r \leftarrow \mathcal{RS}_{\mathsf{LNIZK}}$ and $s \leftarrow \mathcal{RS}_{\mathsf{PKE.Enc}}$, computes $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{apk}, \mathsf{am}; s)$ and $\pi' \leftarrow \mathsf{LNIZK.Prove}(x, w, l = \mathsf{ct}; r)$, sets $\pi = (\pi', \mathsf{ct})$ and $r^* = (r, \mathsf{ct})$, and sends (π, r^*) to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs a bit b'. This is actually the $b = 1$ branch of $\mathsf{G}^{\mathrm{anam}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda)$.

To bound the hop, we construct a PPT IND$-CPA adversary $\mathcal{B}$.

After receiving the pk from its challenger, $\mathcal{B}$ generates $\mathsf{crs}_{\mathsf{LNIZK}} \leftarrow \mathsf{Setup}_{\mathsf{LNIZK}}(1^\lambda)$, then sends $(\mathsf{crs}, \mathsf{apk} := \mathsf{pk})$ to $\mathcal{A}$. $\mathcal{B}$ answers $\mathcal{A}$'s query as follows.

- $\mathcal{O}_{\mathsf{chal}}(x, w, \mathsf{am})$: If $(x, w) \notin \mathcal{R}$ or $\mathsf{am} \notin \mathcal{AM}$, the $\mathcal{B}$ returns $\perp$. Otherwise, it submits am to its challenger of the game of IND$-CPA security of PKE and receives ct'. After that, it samples $r \leftarrow \mathcal{RS}_{\mathsf{LNIZK}}$, computes $\pi' \leftarrow \mathsf{LNIZK.Prove}(x, w, l = \mathsf{ct}'; r)$, $\pi = (\pi', \mathsf{ct}')$, and $r^* = (r, \mathsf{ct}')$. Then it gives (π, r^*) to $\mathcal{A}$.

Finally, when $\mathcal{B}$ receives the final output β from $\mathcal{A}$, it returns $b' = 1 - \beta$ to the challenger of the IND$-CPA game. A uniform ct' perfectly simulates Game 0, whereas an encryption of am perfectly simulates Game 1. Therefore,

$$|\Pr[E_1] - \Pr[E_0]| \leq 2\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathsf{PKE},\mathcal{B}}(\lambda).$$

Since the hidden bit in $\mathsf{G}^{\mathrm{anam}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}$ is uniform,

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{anam}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda) &= |\frac{1}{2}(1 - \Pr[E_0]) + \frac{1}{2}\Pr[E_1] - \frac{1}{2}| \\
&= \frac{1}{2}|\Pr[E_1] - \Pr[E_0]| \\
&\leq \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathsf{PKE},\mathcal{B}}(\lambda) \leq \mathsf{negl}(\lambda).
\end{aligned}$$

Thus, ASY-ANIZK satisfies anamorphic security.

H.2 Proof of Weak Robustness

Proof. Let $\mathcal{A}$ be a PPT adversary in $\mathsf{G}^{\mathrm{wrob}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda)$, and let E denote the event that this game returns 1. After receiving $(\mathsf{crs}, \mathsf{apk})$, $\mathcal{A}$ outputs (x, w). If $(x, w) \notin \mathcal{R}$, the game terminates and E does not occur. Otherwise, the challenger generates an ordinary proof as

$$\rho \leftarrow \mathcal{LS}, \qquad \pi' \leftarrow \mathsf{LNIZK.Prove}(x, w, l = \rho), \qquad \pi = (\pi', \rho).$$

The label ρ is sampled after $\mathcal{A}$ fixes (x, w) and is uniform and independent of the adversary's choice. By the construction of $\mathsf{aExtract}$, we have $\mathsf{PKE.Dec}(\mathsf{ask}, \rho) \neq \perp$. The random ciphertext rejection assumption in Theorem 3 gives exactly the RCR bound stated above. Consequently,

$$\begin{aligned}\mathbf{Adv}^{\mathrm{wrob}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda) &= \Pr[E] \\ &\leq \Pr[\mathsf{PKE.Dec}(\mathsf{ask}, \rho) \neq \perp : \rho \leftarrow \mathcal{LS}] \\ &\leq \mathsf{negl}(\lambda).\end{aligned}$$

Therefore, ASY-ANIZK satisfies weak robustness.

H.3 Proof of Recipient Anamorphic Zero Knowledge

Proof. Let $\mathcal{S}_{\mathsf{LNIZK}} = (\mathcal{S}_{\mathsf{LNIZK}}.\mathsf{Setup}, \mathcal{S}_{\mathsf{LNIZK}}.\mathsf{Prove})$ be the simulator of zero knowledge property of LNIZK, then we construct a recipient anamorphic zero knowledge simulator $\mathcal{S}_{\mathsf{ana}}$ as follows:

$$\begin{aligned}\mathcal{S}_{\mathsf{ana}}.\mathsf{Setup}(1^\lambda): &\quad \text{return } (\mathsf{crs}, \tau) \leftarrow \mathcal{S}_{\mathsf{LNIZK}}.\mathsf{Setup}(1^\lambda). \\ \mathcal{S}_{\mathsf{ana}}.\mathsf{aProve}(\tau, x, \mathsf{apk}, \mathsf{am}): &\quad \mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{apk}, \mathsf{am}), \\ &\quad \pi' \leftarrow \mathcal{S}_{\mathsf{LNIZK}}.\mathsf{Prove}(\tau, x, l = \mathsf{ct}), \quad \text{return } (\pi', \mathsf{ct}).\end{aligned}$$

Assume that there exists a PPT adversary $\mathcal{A}$ can distinguish the two games in Fig. 17 with non-negligible probability. We construct a PPT adversary $\mathcal{B}$, who attempts to break the labeled zero knowledge property of LNIZK.

After $\mathcal{B}$ receives crs from the challenger in the game of the labeled zero knowledge property of LNIZK, it generates $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, and sends $(\mathsf{crs}, \mathsf{apk}, \mathsf{ask})$ to $\mathcal{A}$.

$\mathcal{B}$ answers $\mathcal{A}$'s oracle queries as follows.

- On input (x, w, am), if $(x, w) \notin \mathcal{R}$ or $\mathsf{am} \notin \mathcal{AM}$, it returns $\perp$.
- Otherwise, it samples $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{apk}, \mathsf{am})$, submits $(x, w, l = \mathsf{ct})$ to its real-or-simulated labeled proving oracle, obtains π', and returns (π', ct) to $\mathcal{A}$.

$\mathcal{B}$ finally returns the bit output by $\mathcal{A}$. If its oracle is the real LNIZK prover, then $\mathcal{A}$'s view is exactly that in $\mathsf{G}^{\mathrm{razk}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A}}(\lambda)$. If its oracle is the LNIZK simulator, then the view is exactly that in $\mathsf{G}^{\mathrm{razk}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda)$. Hence,

$$\mathbf{Adv}^{\mathrm{razk}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda) \leq \mathbf{Adv}^{\mathrm{lzk}}_{\mathsf{LNIZK},\mathcal{B},\mathcal{S}_{\mathsf{LNIZK}}}(\lambda) \leq \mathsf{negl}(\lambda).$$

Therefore, ASY-ANIZK satisfies recipient anamorphic zero knowledge.

H.4 Proof of Recipient Anamorphic Simulation Extractability

Proof. Let $\mathcal{S}_{\mathsf{LNIZK}}$ be the simulator with respect to which LNIZK satisfies labeled zero knowledge and simulation extractability, as defined in Fig. 19. We use the simulator $\mathcal{S}_{\mathsf{ana}}$ constructed from this same $\mathcal{S}_{\mathsf{LNIZK}}$ in Appendix H.3. As in Definition 31, crs is an implicit input to the relevant algorithms.

Fix any PPT adversary $\mathcal{A}$ in the RASE game of Fig. 18. We first construct a PPT adversary $\mathcal{B}$ for the simulation extractability game of LNIZK. After receiving a simulated crs from its challenger, $\mathcal{B}$ independently generates $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, initializes $Q_{\mathsf{ana}} := \emptyset$, and gives $(\mathsf{crs}, \mathsf{apk}, \mathsf{ask})$ to $\mathcal{A}$. For clarity, we write $\mathcal{O}_{\mathsf{Sim}}^{\mathsf{LNIZK}}$ for $\mathcal{B}$'s simulation oracle in Fig. 19. On each query (x, am) to $\mathcal{A}$'s oracle $\mathcal{O}_{\mathsf{Sim}}$, $\mathcal{B}$ proceeds as follows.

- If $\mathsf{am} \notin \mathcal{AM}$, it returns $\perp$.
- Otherwise, it samples $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{apk}, \mathsf{am})$ with fresh randomness and submits $(x, l = \mathsf{ct})$ to $\mathcal{O}_{\mathsf{Sim}}^{\mathsf{LNIZK}}$.
- On receiving π', it sets $\pi = (\pi', \mathsf{ct})$, inserts (x, π) into Q_{ana}, and returns π to $\mathcal{A}$.

When $\mathcal{A}$ outputs (x^*, π^*), $\mathcal{B}$ parses $\pi^* = (\pi'^*, \mathsf{ct}^*)$ and outputs $(x^*, l^* = \mathsf{ct}^*, \pi'^*)$ to its challenger.

By the simulation extractability of LNIZK, there exists a PPT extractor $\mathsf{Ext}_{\mathsf{LNIZK}}^{\mathcal{B}}$ for the adversary $\mathcal{B}$ fixed above such that its winning probability in Fig. 19 is negligible. We now define the extractor for $\mathcal{A}$ as follows:

$$\mathsf{Ext}_{\mathsf{ASY\text{-}ANIZK}}^{\mathcal{A}}(\tau, \mathsf{apk}, \mathsf{ask}, x^*, \pi^*) :$$
$$\text{Parse } \pi^* = (\pi'^*, \mathsf{ct}^*), \quad \text{Return } \mathsf{Ext}_{\mathsf{LNIZK}}^{\mathcal{B}}(\tau, x^*, l^* = \mathsf{ct}^*, \pi'^*).$$

The inputs apk and ask are included to match the extractor interface in Fig. 18, although this construction does not use them. The extractor is PPT and requires only its stated inputs and the implicit crs. In particular, it does not require a query transcript or $\mathcal{A}$'s internal state. The challenger invokes this extractor after receiving $\mathcal{A}$'s output.

We now bound the winning probability of $\mathcal{A}$. Let E denote the event that $\mathcal{A}$ wins $\mathsf{G}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}^{\mathrm{rase}}(\lambda)$ when the challenger uses $\mathsf{Ext}_{\mathsf{ASY\text{-}ANIZK}}^{\mathcal{A}}$ defined above. By Fig. 18, this event occurs precisely when

$$\mathsf{Verify}(x^*, \pi^*) = 1, \qquad (x^*, \pi^*) \notin Q_{\mathsf{ana}}, \qquad (x^*, w^*) \notin \mathcal{R},$$

where $w^* \leftarrow \mathsf{Ext}_{\mathsf{ASY\text{-}ANIZK}}^{\mathcal{A}}(\tau, \mathsf{apk}, \mathsf{ask}, x^*, \pi^*)$.

The simulation provided by $\mathcal{B}$ gives $\mathcal{A}$ exactly the view prescribed in Fig. 18. Indeed, the simulated crs and the independently generated $(\mathsf{apk}, \mathsf{ask})$ have the required distributions. For each query (x, am) with $\mathsf{am} \in \mathcal{AM}$, $\mathcal{B}$ returns $\pi = (\pi', \mathsf{ct})$, where $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{apk}, \mathsf{am})$ and $\pi' \leftarrow \mathcal{S}_{\mathsf{LNIZK}}.\mathsf{Prove}(\tau, x, l = \mathsf{ct})$. This is exactly the output distribution of $\mathcal{S}_{\mathsf{ana}}.\mathsf{aProve}(\tau, x, \mathsf{apk}, \mathsf{am})$. Queries with invalid messages receive $\perp$, as required. Thus, this simulation preserves the winning probability of $\mathcal{A}$, including when its queries are adaptive and it uses ask.

Let Q_{LNIZK} denote the set recorded by $\mathcal{B}$'s challenger in Fig. 19. Every query made by $\mathcal{B}$ to $\mathcal{O}_{\mathsf{Sim}}^{\mathsf{LNIZK}}$ corresponds to one simulated proof returned to $\mathcal{A}$, and $\mathcal{B}$ makes no other queries to this oracle. Consequently,

$$Q_{\mathsf{LNIZK}} = \big\{(x, \mathsf{ct}, \pi') : (x, (\pi', \mathsf{ct})) \in Q_{\mathsf{ana}}\big\}.$$

This equality gives, for every well-formed output $\pi^* = (\pi'^*, \mathsf{ct}^*)$,

$$(x^*, \pi^*) \notin Q_{\mathsf{ana}} \quad \Longleftrightarrow \quad (x^*, \mathsf{ct}^*, \pi'^*) \notin Q_{\mathsf{LNIZK}}.$$

The correspondence uses the full tuple (x, l, π') recorded in Fig. 19. Thus, it also covers an output that reuses a previously queried statement or label but changes another component of the tuple.

By the construction in Fig. 9,

$$\mathsf{Verify}(x^*, \pi^*) = \mathsf{LNIZK.Verify}(x^*, \pi'^*, l = \mathsf{ct}^*).$$

Moreover, by the definition of $\mathsf{Ext}^{\mathcal{A}}_{\mathsf{ASY\text{-}ANIZK}}$, the RASE challenger and $\mathcal{B}$'s challenger invoke $\mathsf{Ext}^{\mathcal{B}}_{\mathsf{LNIZK}}$ on the same input $(\tau, x^*, \mathsf{ct}^*, \pi'^*)$. Taking the same random coins for the corresponding algorithms, including the extractor, yields the same w^*. Hence, whenever E occurs, the output $(x^*, l^* = \mathsf{ct}^*, \pi'^*)$ of $\mathcal{B}$ is accepting and fresh, and the extracted witness satisfies $(x^*, w^*) \notin \mathcal{R}$. These are exactly the winning conditions in Fig. 19. Conversely, each of these conditions implies its corresponding RASE condition. The two winning probabilities are therefore equal, and we obtain

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{rase}}_{\mathsf{ASY\text{-}ANIZK},\mathcal{A},\mathcal{S}_{\mathsf{ana}}}(\lambda) &= \Pr[E] \\
&= \Pr[\mathsf{G}^{\mathrm{se}}_{\mathsf{LNIZK},\mathcal{B},\mathcal{S}_{\mathsf{LNIZK}}}(\lambda) = 1] \\
&= \mathbf{Adv}^{\mathrm{se}}_{\mathsf{LNIZK},\mathcal{B},\mathcal{S}_{\mathsf{LNIZK}}}(\lambda) \leq \mathsf{negl}(\lambda).
\end{aligned}$$

Since $\mathcal{A}$ was arbitrary and $\mathsf{Ext}^{\mathcal{A}}_{\mathsf{ASY\text{-}ANIZK}}$ is PPT, ASY-ANIZK satisfies recipient anamorphic simulation extractability.